Received: 16 September 2023 | Accepted: 26 April 2024

DOI: 10.1111/bjet.13482

ORIGINAL ARTICLE

British Journal of Educational Technology BERA

Enhancing computational thinking and spatial reasoning skills in gamification programming learning: A comparative study of tangible, block and paper-and-pencil tools

Xin Gong[1] | Weiqi Xu[2] | Shufan Yu[3] | Jingjing Ma[3] | Ailing Qiao[1]

[1]College of Education, Capital Normal University, Beijing, China

[2]College of Education, Zhejiang University, Hangzhou, China

[3]School of Educational Information Technology, Faculty of Artificial Intelligence in Education, Central China Normal University, Wuhan, Hubei, China

Correspondence
Ailing Qiao, College of Education, Capital Normal University, 105 West Third Ring North Road, Haidian District, Beijing 100048, China.
Email: qiaoal@126.com

Funding information
Key Project of Education Sciences Research of Chinese Ministry of Education in 2022, Grant/Award Number: DCA220449

Abstract

Tangible programming tools have become a mainstream teaching aid in gamification programming learning (GPL) due to their interactivity and ability to enhance novice learners' computational thinking and spatial reasoning skills. However, comparing the relative efficacy of different programming tools that simultaneously support these skills was not adequately explored. This study designed and evaluated three programming tools: the tangible programming tool (TPG), which uses real touchable objects; the block programming tool (BPG), which employs virtual programming blocks and 3D game scenarios; and the paper-and-pencil programming tool (PPG), which uses paper and pen to draw. The study involved 112 seventh-grade students from three natural classes: Class A (TPG, n_1=37), Class B (BPG, n_2=38), and Class C (PPG, n_3=37). These students completed four gamification programming tasks and CT skills, spatial reasoning skills, enjoyment, cognitive load and GPL task list measurements. The results indicated that the tangible programming tool led to lower cognitive load, significant improvement in spatial reasoning skills and better abstraction and problem decomposition skills. The block programming tool provided a more enjoyable experience and facilitated students' algorithm design and efficiency. The paper-and-pencil programming tool was found to be less effective in improving spatial reasoning skills. This study's findings can help programming educators cultivate students' thinking skills and improve their learning experience by effectively selecting the most appropriate programming tools.

KEYWORDS
computational thinking, gamification programming learning, programming tools, spatial reasoning skills

Practitioner notes

What is already known about this topic

- Combining gamification with programming can increase interest, promote engagement, and improve learning outcomes for novice programmers.
- Some existing programming tools differ in developing students' thinking skills and optimizing their learning experience.

What this paper adds

- The exercises "Correspondence between three-view drawing and stereogram" and "Programming mazes" can be used to hone both spatial reasoning and computational thinking skills.
- The tangible programming tool was beneficial for improving students' spatial reasoning skills because it went beyond the flatness of previous programming tools and offered embodiment and interactivity.
- The paper-and-pencil programming tool was found to be less effective in improving spatial reasoning skills due to its time-consuming stereogram-building process.
- The block programming tool provided a more enjoyable experience due to its 3D immersion and ease of operation, facilitating students' algorithm design and efficiency.

Implications for practice and/or policy

- Well-designed programming tools can reduce the cognitive load on students during gamification programming learning to assist them in constructing their thinking systems.
- Educators should select the most appropriate tools and incorporate game mechanics such as competition to meet the maximum progress of students' target skills.

INTRODUCTION

With the rapid development of computer technologies in the 21st century, programming learning has become an essential field in education. Learning programming is considered to be a tedious and challenging task for novice programmers as they need to understand abstract concepts like functions and master various skills, such as problem-solving, program design, and debugging, at the same time (Garcia, 2021). Gamification programming learning (GPL) refers to organizing game activities and creating a gamification environment for students in programming education (Hamari et al., 2016), which has the potential to maintain students' learning interest and promote their engagement (Zhan et al., 2022). Furthermore, some different programming tools (eg, paper-and-pencil, block, tangible) encapsulate abstract concepts to help novice programmers learn programming concepts (Chiu & Huang, 2015), develop computational thinking (CT) (Lee et al., 2021) and foster problem-solving ability (Liu et al., 2022). Research has shown that the three programming tools were used to implement GPL in different operational forms. First, paper-and-pencil programming tools allow students to use paper and pen to draw meaningful symbols during GPL (Kim et al., 2013). Second, block programming tools visualize coding by dragging program blocks or icons on the computer screen (Sapounidis & Demetriadis, 2013). Third, tangible programming tools use unplugged blocks that are similar to jigsaw puzzles or Lego blocks for sequencing to achieve programming (Song, 2019).

Previous research has found that the use of different programming tools had distinct effects on students' thinking skills and learning experiences during GPL. In terms of thinking skills, the effects on CT and spatial reasoning skills were explored. Studies have found that students have difficulty perceiving the spatial motion of robots when using modular robot programming tools to develop CT skills (Angeli & Valanides, 2020; Fessakis et al., 2013), mainly due to a lack of high-level spatial reasoning skills, it can be seen that students' spatial reasoning skills are correlated with CT skills. Some studies also suggest that tangible programming tools are more helpful in developing CT and spatial reasoning skills (Misirli et al., 2019). However, there is a lack of empirical studies on the differences between it and other programming tools regarding the finer dimensions of developing CT and spatial reasoning skills. In terms of learning experience, research centred primarily around enjoyment and cognitive load. Although no significant differences in enjoyment were found across programming tools (Xie et al., 2008), students showed lower cognitive load when using block and tangible programming tools (Huang et al., 2023). To fill these gaps and provide insights into the optimal designs of GPL environments, this study designed three programming tools (paper-and-pencil, block and tangible) and further explored the effects of different programming tools on students' thinking skills (ie, CT skills and spatial reasoning skills) and learning experience (ie, enjoyment and cognitive load).

LITERATURE REVIEW

Gamification programming learning

Programming learning is an educational form that allows students to creatively solve practical problems using the programming operation's thinking modes. It has been widely used in various educational contexts, such as K-12 and higher education, to cultivate students' thinking skills (Sun et al., 2021), improve their computer science career interests (Chittum et al., 2017) and foster their motivation and achievement (Erol & Kurt, 2017). To enhance the learning interest of novice programmers, gamification strategies are integrated into programming learning. The potential of gamification in the programming education domain is based on the use of its mechanisms and elements. GPL highlights the use of game mechanisms (eg, competitions and rewards) that represent intrinsic motivation, as well as game elements (eg, points and feedback) that represent extrinsic motivation, aiming to enhance the thinking skills and learning experience of novice programmers (Lindberg et al., 2018; Rojas-López et al., 2019). For instance, Barradas et al. (2020) employed game mechanisms such as rewards to encourage students to program in order to build CT skills based on a virtual block programming environment. Carreño-León et al. (2018) chose game elements such as cards to allow students to programme using physical algorithmic cards, which enhanced their enjoyment and cognitive ability when solving complex problems. Additionally, gamification has been found to contribute to student engagement in programming courses in studies conducted by Figueiredo and Garcia-Penalvo (2020) and Quevedo Gutierrez and Zapatera Llinares (2021). Earlier studies have shown using different programming tools in GPL for gamification activities can help develop students' thinking skills. However, there is a clear gap in comparing these tools to understand their unique effects.

Programming tools in GPL

Programming tools are a carrier for creating complex interactions to achieve problem-solving goals in the programming learning process. In order to maintain students' interest, enhance their academic performance and develop their higher-order thinking skills, different types of programming tools have been applied to GPL, such as paper-and-pencil programming tools (Kim et al., 2013), block programming tools (Hu et al., 2021), and tangible programming tools (Wang et al., 2024). The operational modes of different programming tools exhibit significant differences, allowing various programming types to organize educational gaming activities based on the characteristics of their respective tools.

Specifically, first, paper-and-pencil programming involves transforming specific issues into logical representations in programming languages using paper, pen and special symbols (Yildiz Durak, 2020). It exemplifies gamification by creating a competitive environment where two programmers compete in matches to solve challenges on a chessboard. According to Kim et al. (2013), using only paper and pen as programming tools significantly improved students' understanding of CT concepts. For instance, the board programming game leverages pencil and paper to adeptly translate abstract logical reasoning from the mind onto paper through symbol drawing, effectively converting mental models into structured logical representations in solving game challenges (Robert et al., 2022).

Second, block programming employs visual elements and a user-friendly format to reduce syntax errors, enabling the creation of programs through the drag-and-drop of visual objects on the screen using a mouse (Dilmen et al., 2023). Scratch is an educational tool based on visual programming blocks (Dúo-Terrón, 2023). ScratchJr uses icons instead of text to simplify code presentation (Strawhacker et al., 2018). For example, these tools leverage the interactive gaming elements of the programming platform to provide students with a gamification learning environment and allow them to experience the joy of designing digital games by programming to create mini-games (Sapounidis et al., 2015).

Third, tangible programming uses building blocks and plug-in robots as carriers, motion buttons and instruction cards as drivers to enable tangible entities to complete specific action tasks. For example, KIBO (Bers et al., 2019), Beebot Robots (Angeli & Valanides, 2020) and Cubetto (Pérez-Marín et al., 2022) are programmed through interaction with physical objects or cards, providing a hands-on programming experience. The main gamification presentation of tangible programming is the creation of 'three-dimensional mazes' that allow students to synthesize the robot's path and quickly respond to simple programming instructions such as going straight or turning, facilitating students' thinking training and helping them recognize simple geometric shapes. The research conducted by Strawhacker and Bers (2015) suggested that the tangible programming tool was more appropriate for younger students with weak computer skills, as it helped them build their virtual worlds and matched their developmental thinking skills.

However, previous studies have revealed the potential contributions of these three programming tools (ie, paper and pencil programming tools, block programming tools and tangible programming tools), but there is still a lack of systematic comparison of how these tools relate effects to students' GPL. Therefore, it is necessary to further investigate the effects of different programming tools on students' learning outcomes during their GPL, and this helps us choose the most suitable programming tool to learn the target skills.

Effects of different programming tools on students' GPL

GPL cultivates students' thinking skills through the creation of programming activities while enhancing the learning experience through the incorporation of gamification elements and mechanisms. Recently, some studies have found that different programming tools have varied effects on students' GPL, especially their thinking skills and learning experience.

First, from the perspective of thinking skills, CT and spatial reasoning skills are the main focus of cultivation in GPL, as they are intertwined in their utilization of spatial data and objects for problem-solving (Città et al., 2019; Román-González et al., 2017). Spatial reasoning skills help students navigate programming objects correctly, while CT skills transform the navigation process into a problem-solving framework through programming (Clarke-Midura et al., 2021). Specifically, CT skills refer to the ability to apply abstract concepts from computer science to solve real-world problems and typically encompass abstraction (eg, extracting the essence of a complex system), problem decomposition (eg, breaking down a complex problem into manageable parts), algorithm design (constructing an ordered series of steps to solve a problem) and algorithm efficiency (solving a problem by designing a minimum number of steps and removing redundant and unnecessary steps) (Ou-Yang et al., 2023). Previous research has found different effects of different programming tools on learners' CT skills development in GPL. For example, Ou-Yang et al. (2023) reported that learners using AR Bot had significantly higher scores in algorithm design and algorithm efficiency than Scratch by comparing the effects of AR Bot and Scratch on CT skills. In addition, Spatial Reasoning skills involve the ability to comprehend, reason, recall and manipulate relationships between objects or spaces and are seen as a collection of different abilities (Hegarty et al., 2006) that often include mental rotation, spatial orientation and spatial visualization (Ramful et al., 2016). Previous research has found different effects of different programming tools on learners' development of spatial reasoning skills in gamification programming. For example, Antle (2013) compared the effects of tangible, physical and mouse-based interaction styles on students' spatial problem-solving tasks and found that tangible may support the development of thinking skills through hands-on child-computer interactions.

Second, from the perspective of the learning experience, enjoyment and cognitive load are usually highly correlated with students' learning experience (Sharma et al., 2019). Specifically, enjoyment refers to the associated positive emotions and feelings of pleasure, which is an essential variable in the learning process (Jordan, 2002). Previous research has extensively elaborated that enjoyment was one of the main drivers of GPL (Cheng et al., 2023). For example, Ou-Yang et al. (2023) assessed the effects of AR Bot and Scratch programming tools on students' internal learning processes during a maze game and found that students who used AR Bot showed higher enjoyment of learning. Melcer and Isbister (2017) compared how mouse and tangible input methods affect the learning process based on educational programming games and found that tangibles had a more significant positive impact on situational interest and enjoyment because of more immediate feedback. In addition, cognitive load refers to the load imposed on an individual's cognitive system when performing a specific job (Sweller et al., 1998). In GPL, excessive use of game elements can lead learners to interact with too much irrelevant multimedia (mental load), causing unnecessary stress (mental effort) or new cognitive demands to increase the level of cognitive load (Shaban et al., 2021), which in turn affects learning outcomes (Wang, 2023) Previous studies have assessed the effect of using different programming tools in GPL on students' cognitive load and found that the relationship between learning performance and cognitive load varies with the tool. For example, Huang et al. (2023) integrated augmented reality (AR) technology into a programming game and found that the cognitive load of the AR board game group was lower than that of the block-based programming group. Zhong et al. (2022) did not find a difference in the cognitive load of the programming learning process between tangible programming (Boson Kits) and the graphical programming tool (CFunWorld AS Block).

Overall, there is a difference in the literature when it comes to investigating how the different programming tools affect students' thinking skills and learning experience in GPL. More importantly, most of them only compared the differences between the two tools, still largely insufficient to judge the relative effectiveness of existing popular programming tools. Therefore, it is essential to explore the effects of different programming tools on students' thinking skills (ie, CT skills and spatial reasoning skills) and learning experience (ie, enjoyment and cognitive load) during GPL to fill the gaps in the current research.

Purposes of this study

Although previous research has highlighted the potential benefits of using different programming tools in GPL to enhance students' thinking skills, existing research has not adequately compared the effects of different programming tools in GPL on cultivating students' thinking skills and enhancing their learning experience. Therefore, to provide valuable insights for selecting the most suitable programming tool for enhancing specific target skills in GPL, we designed three programming tools to support middle school students' GPL, including tangible (ie, physical-based), block (mouse-based) and paper-and-pencil. Specifically, we adopted a quasi-experimental study to examine three programming tools' effects on students' thinking skills (ie, CT and spatial reasoning skills) and learning experience (ie, enjoyment and cognitive load) during GPL. Two research questions were proposed:

RQ1. How do the three programming tools affect students' thinking skills (ie, CT and spatial reasoning skills) in GPL?
RQ2. How do the three programming tools affect students' learning experience (ie, enjoyment and cognitive load) in GPL?

METHODOLOGY

Participants and research context

This research involved 112 students, aged between 12 and 14 years ($M=13.071$, $SD=0.549$), who were studying in the seventh-grade at an urban public middle school in northern China. We randomly selected three classes and assigned them directly to three experimental groups. Class A, namely *tangible programming* group (TPG), with 37 students, Class B, namely *block programming* group (BPG), with 38 students, and Class C, namely p*aper-and-pencil programming* group (PPG), with 37 students. Ethical approval was obtained before the research, and all participants agreed to participate and submitted parental consent.

Research tools design

In this research, to explore the differential impact of different programming tools in the GPL, we designed four GPL task lists (shown in Table 1) and a series of GPL tools, namely *magic*

TABLE 1 GPL task lists content.

Theme	Content	CT skills
'?' Word Maze	(1) Based on a three-view drawing, use nine building blocks to build '?' Word Maze (2) Based on the shape characteristics of the maze, understand the concept of 'Sequential' and use code cards for sequencing the robot's movement commands to help the robot find the treasure	(1) Abstraction: Understanding the relationship between the 'action' command on the code card and the action performed by the robot (2) Problem decomposition (3) Algorithm design (sequences): Designing the robot's sequential forward path (4) Algorithm efficiency
'◎' Word Maze	(1) Based on a three-view drawing, use 35 building blocks to build '◎' word maze (2) Based on the shape characteristics of the maze, understand the concept of 'repetitive' and use the loop code cards to help the robot collect all the treasures as quickly as possible (Choose the most appropriate movement path for the current situation)	(1) Abstraction: Understanding the relationship between the 'function' commands on the code cards and the actions performed by the robot (2) Problem decomposition (3) Algorithm design (loops): Identifying patterns within robot paths, outlining the patterns of repeated commands in the loop, and designing the robot's path forward (4) Algorithm efficiency: For a faster collection of all the treasures, you can use functions under suitable conditions to simplify the program logically by encapsulating specific steps
√" Word Maze	(1) Based on a three-view drawing, use six building blocks to build '√' word maze (2) Based on the shape characteristics of the maze, understand the concept of 'conditional' and use colouring code cards to help the robot make choices	(1) Abstraction: Understanding the relationship between the 'conditional' commands on the code cards and the actions performed by the robot (2) Problem Decomposition: to break down a problem into smaller components and to analyse the solution steps of problems (3) Algorithm design (conditionals): Judging whether the colour of the robot is changed after the action command is executed. If yes, only the treasure with the same colour as the robot will be searched (4) Algorithm efficiency: Using fewer code cards to achieve the goal
Challenge	(1) Every student designs the stereogram and specifies the CT concept included in the task (2) If your opponent gets the question right, your opponent gets a star; if your opponent doesn't, you get a star	

block, with three sets of programming tools (ie, tangible, block, and paper-and-pencil programming tools) (shown in Figure 1). In addition, we also incorporated standard features of existing gamification designs (eg, students program a robot to solve a puzzle), which provided a point of comparison between different programming tools.

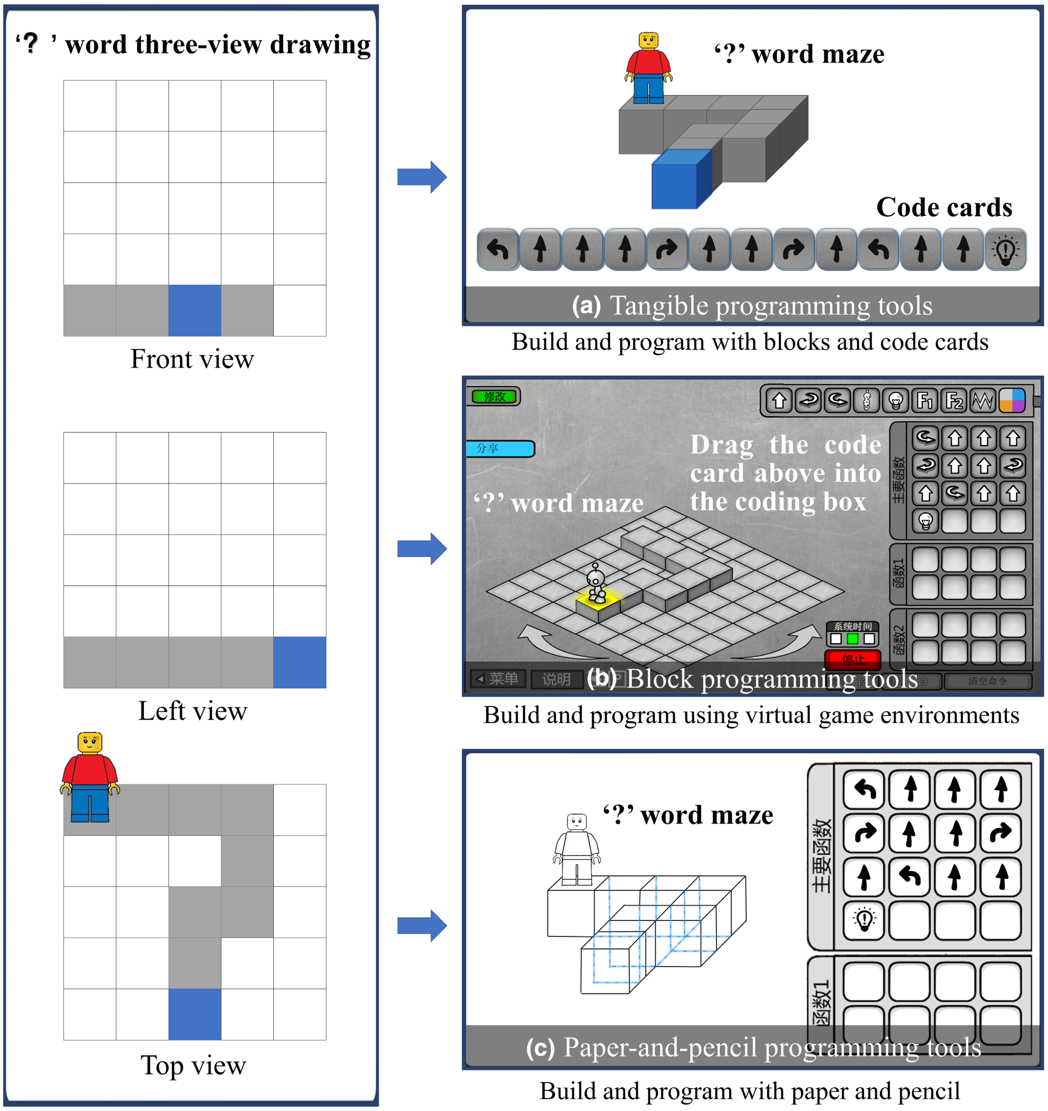
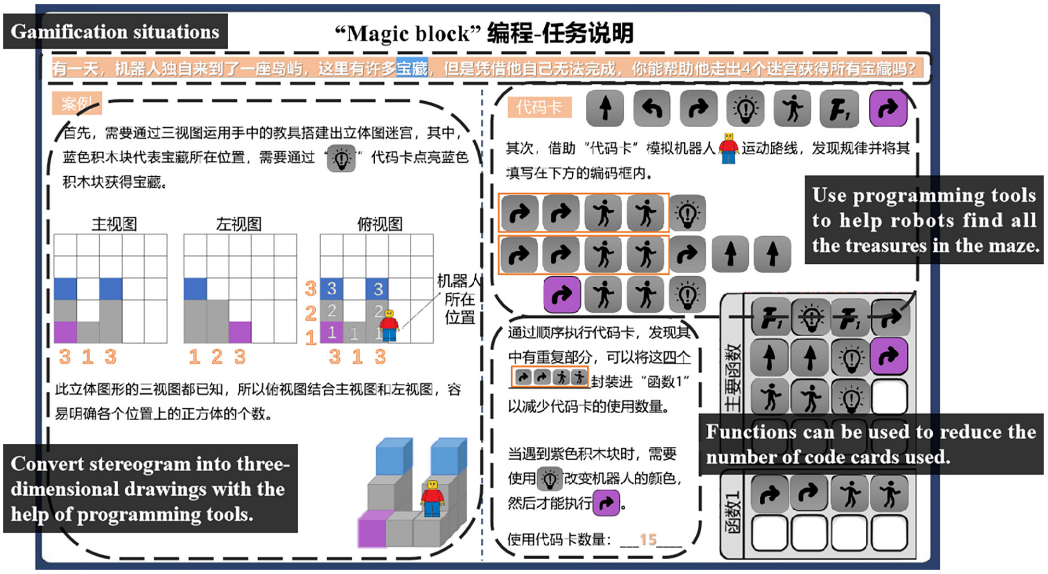

'? ' word three-view drawing
Front view
Left view
Top view
'?' word maze
Code cards
(a) Tangible programming tools
Build and program with blocks and code cards
修改
分享
Drag the code card above into the coding box
'?' word maze
主要函数
函数1
函数2
系统时间
停止
菜单
说明
清空命令
(b) Block programming tools
Build and program using virtual game environments
'?' word maze
主要函数
函数1
(c) Paper-and-pencil programming tools
Build and program with paper and pencil

FIGURE 1 The design of three programming tools in GPL. (a) Tangible programming tool. (b) Block programming tool. (c) Paper-and-pencil programming tool.

GPL task lists

We designed four GPL task lists, each containing two subtasks. Subtask 1 was 'Correspondence between three-view drawing and stereogram', and subtask 2 was 'Find the treasure'. The details are shown in Table 1.

The physical drawing of the task list is shown in Figure 2. First, the construction of a stereogram was a two-dimensional to three-dimensional spatial organization, and students should imagine that their position changed while keeping the relative position of the internal parts of the spatial object unchanged (spatial orientation and mental rotation). Second, establish relationships with the robot's actions using code cards (abstraction) and build a sequence of sequential steps (problem decomposition) to solve the problem (algorithm design) (Ou-Yang et al., 2023). In addition, to collect all the treasures as soon as possible, functions


Gamification situations
主视图
左视图
俯视图
机器人所在位置
3 1 3
1 2 3
3 1 3
代码卡
主要函数
函数1
Use programming tools to help robots find all the treasures in the maze.
Functions can be used to reduce the number of code cards used.
Convert stereogram into three-dimensional drawings with the help of programming tools.

"Magic block" 编程-任务说明

有一天，机器人独自来到了一座岛屿，这里有许多宝藏，但是凭借他自己无法完成，你能帮助他走出4个迷宫获得所有宝藏吗?

案例

首先，需要通过三视图运用手中的教具搭建出立体图迷宫，其中，蓝色积木块代表宝藏所在位置，需要通过"💡"代码卡点亮蓝色积木块获得宝藏。

此立体图形的三视图都已知，所以俯视图结合主视图和左视图，容易明确各个位置上的正方体的个数。

其次，借助"代码卡"模拟机器人运动路线，发现规律并将其填写在下方的编码框内。

通过顺序执行代码卡，发现其中有重复部分，可以将这四个封装进"函数1"以减少代码卡的使用数量。

当遇到紫色积木块时，需要使用💡改变机器人的颜色，然后才能执行。

使用代码卡数量：__15__

FIGURE 2 The physical drawing of the GPL task list.

will be used under suitable conditions to achieve logical simplification of the program by encapsulating specific steps (algorithm efficiency). Third, the process of robot movement was in line with the process of algorithm execution, which inspired students to observe 3D objects from multiple perspectives and to perceive 3D objects from different positions and directions in a natural environment (spatial visualization, spatial perception).

GPL tools

We designed tangible, block and paper-and-pencil programming tools to investigate their impact on students' thinking skills and learning experience in GPL, which is more effective.

Tangible programming tool

The tangible programming tools consisted of magnet blocks with specific functions that could be magnetized into various shapes to assemble an extensive three-dimensional route, physical code cards with directional arrows or action symbols, and a robot, as shown in Figure 1a. Students in the tangible programming group were asked to use magnet blocks with different functions to build a stereogram based on the two-dimensional three-view drawing (front view, top view and left view) in the task list and then manually controlled the robot to find the treasure by coding the physical code cards, specific see Figure 3a.

Block programming tool

The block programming tools, as shown in Figure 1b, were implemented with the help of the 'level editor' of Lightbot 2.0, and the rules of use were the same as those of the tangible programming tools, with the core difference that the magnet blocks and physical code cards in the tangible programming tools were transformed into 3D virtual tools (displayed on the computer screen), and the GPL tasks were completed in the 3D virtual environment. Students in the block programming group were asked to drag and drop 3D virtual blocks from the game to build a stereogram and then code to help the robot find the treasure by dragging and dropping virtual code cards from the programming box, specifically see Figure 3b.

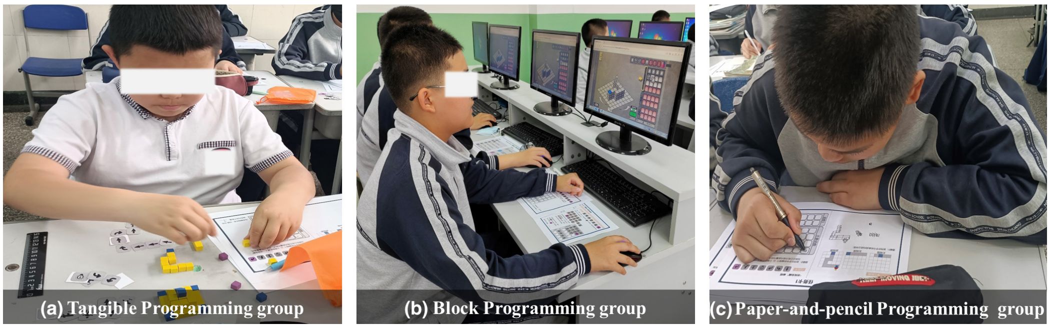

(a) Tangible Programming group
(b) Block Programming group
(c) Paper-and-pencil Programming group

FIGURE 3 The application of GPL tools. (a) Tangible programming tool. (b) Block programming tool. (c) Paper-and-pencil programming tool.

Paper-and-pencil programming tool

The paper-and-pencil programming tools are shown in Figure 1c. The tasks of the paper-and-pencil programming were precisely replicated from the tangible programming. The main difference between the two tools was that the tangible programming tools were used to solve the problem by building a stereogram and using physical code cards to make the robot move, whereas the paper-and-pencil programming tools were used to draw a stereogram on a square map and to design algorithm using the same code symbolic as the physical code cards. The blocks of the tangible programming group correspond to the icons used in the paper-and-pencil programming group, and the computational concepts are mapped similarly. Students in the paper-and-pencil programming group were asked to draw a stereogram on a task list and sequentially draw code symbols to simulate the robot's treasure hunt route through a three-dimensional maze, specifically see Figure 3c.

Research procedure

The research procedure involved four steps, which lasted about 150 minutes (see Figure 4). Before the formal experiment, students were required to fill in the CT and spatial reasoning skills pre-test questionnaire within 30 minutes. Then, all groups received 20 minutes of training on programming tools using rules. During the experiment, each group was given 60 minutes to complete four tasks using different GPL tools, with a consistent learning environment and contents. The teacher provided assistance with issues unrelated to the content without modifying the fundamental elements of the tasks (eg, computer malfunctions or lack of experimental resources). After the experiment, the three groups were given 40 minutes to complete the CT skills, spatial reasoning skills, enjoyment and cognitive load post-tests.

Measurement

We compared the differences between the impact of different GPL tools on students' thinking skills and learning experience.

The measurement of thinking skills

CT skills

We adapted 12 questions from the Bebras tasks (2019 and 2020) as a measurement tool for middle school students' CT skills (abstraction, algorithmic thinking, problem decomposition)

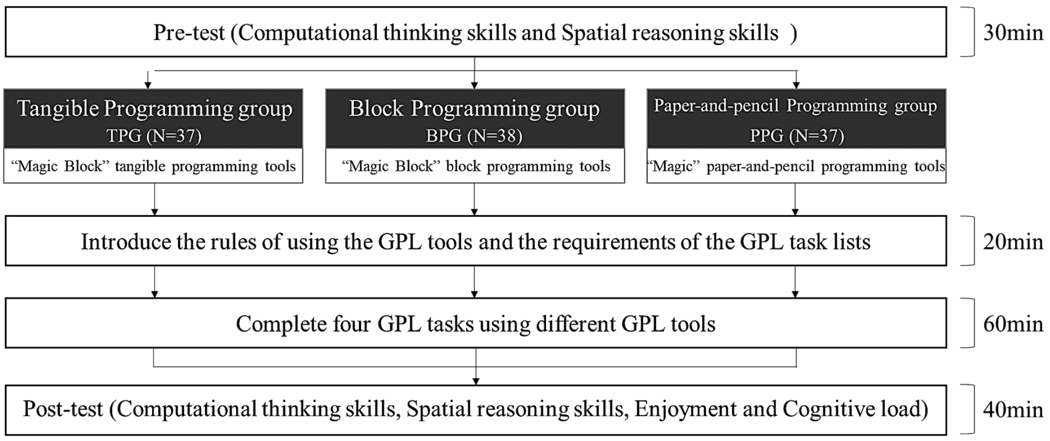

Pre-test (Computational thinking skills and Spatial reasoning skills)
30min
Tangible Programming group
TPG (N=37)
"Magic Block" tangible programming tools
Block Programming group
BPG (N=38)
"Magic Block" block programming tools
Paper-and-pencil Programming group
PPG (N=37)
"Magic" paper-and-pencil programming tools
Introduce the rules of using the GPL tools and the requirements of the GPL task lists
20min
Complete four GPL tasks using different GPL tools
60min
Post-test (Computational thinking skills, Spatial reasoning skills, Enjoyment and Cognitive load)
40min

FIGURE 4 Research process.

(Dagienė & Sentance, 2016). The pre-test and post-test both had 2 easy (3 points each), 2 medium (5 points each) and 2 difficult (7 points each) for a total of 30 points. The Bebras measurement differed from others because it did not require prior knowledge, making it suitable for individuals without programming experience. The scores in the original measurement were coded as 1 for correct answers and 0 for incorrect answers, and the Kuder–Richardson reliability coefficient (KR-20) analysis showed that the pre-test was 0.675 and the post-test was 0.641, indicating the measurement's acceptable consistency. As a general opinion, a KR-20 score>0.50 is regarded as indicative of acceptable reliability (Souza et al., 2017).

Spatial reasoning skills

We chose the standardized test (Ramful et al., 2016) to measure spatial reasoning skills, which contained three sub-dimensions (mental rotation, spatial orientation, and spatial visualization), each consisting of 10 multiple-choice items (30 items in total). We scored 1 point for a correct answer and 0 for an incorrect answer. The KR-20 analysis showed that the pre-test was 0.626 and the post-test was 0.635, indicating the measurement's moderate reliability.

GPL task lists

To make the conclusions in this study more reliable, we used both the CT skills and spatial reasoning skills measurement while also developing scoring standards for the GPL tasks, aiming to ensure stable content validity through multidimensional cross-validation of the data. Based on the specific use of CT skills and spatial reasoning skills in solving GPL tasks, scoring standards (shown in Table 2) were first developed by two Ph.D. students in the field of programming education and then reviewed and revised by the professor who is an expert in the field to form scoring standards applicable to GPL tasks. Two raters who have received training on the rules for using scoring standards independently assessed each student based on tasks reflecting abstraction, problem decomposition, algorithm design, algorithm efficiency and **s**patial reasoning skills (as shown in Table 1), where 1 represents the lowest possible rating, and 5 represents the highest possible rating. Spearman's correlation analysis was utilized to assess rate agreement, yielding correlation coefficients of 0.987, 0.983, 0.981, 1.00 and 0.939 for the five sub-dimensions, all indicating high-rate agreement ($p < 0.001$). After completing the scoring, they cross-checked and discussed each other's rating results, addressed objections regarding the scores for the 29 ratings, and agreed upon the final analysis.

TABLE 2 Scoring standards for GPL tasks.

Dimension	Sub-dimensions	Description	Score
Computational thinking skills	Abstraction	Ability to link 'actions', 'functions' and 'conditional statements' commands on code cards to the actions performed by the robot	5
	Problem decomposition	Ability to deconstruct programming problems into more minor problems	5
	Algorithm design	Ability to develop and implement the ultimate goal of a robot program in which students use functions, conditional statements, etc.	5
	Algorithm efficiency	The score is higher when the student uses fewer blocks	5
Spatial reasoning skills		Correctly translate the three-view drawing into the corresponding stereogram	5

The measurement of the learning experience

Enjoyment

We selected the enjoyment dimension from the Mindful Flow Experience Scale (Pearce et al., 2005) to continuously detect the learners' experiences using programming tools in GPL. It included 'I found the activities enjoyable', 'I found the activities interesting', 'I was frustrated by what I was doing', 'The activities excited my curiosity' and 'The activities bored me'. A Likert scale of 5 was used ('5' for strongly agree and '1' for strongly disagree). Among them, the last question had served as a reverse scoring question to assess the subjects' attentiveness and sincerity in answering the questions. Exploratory factor analysis was utilized to analyse the construct validity of the measurement, successfully identifying an 'Enjoyment' dimension consistent with theoretical expectations after extracting factors with eigenvalues greater than 1 and applying Varimax rotation. Moreover, the Cronbach alpha of the measurement was 0.801, indicating a high confidence level.

Cognitive load

We referred to the cognitive load measurement of Paas et al. (2003) and designed two cognitive load questions consisting of two dimensions, 'mental load' and 'mental effort'. Specifically, the mental load is related to the amount and degree of information interaction between the task and the subject. Mental effort relates to how information is presented and the strategies used to instruct students (Paas & Van Merriënboer, 1994). The item of mental effort was 'What level of effort did I put into the robotics activity'? and the item of mental load was 'How difficult was the task of the robot activity'? The Likert seven-point rating was used, in which 1 meant very easy, 4 meant moderately difficult and 7 meant very difficult. The content validity of the scale was verified through expert review by seven specialists and pre-testing, resulting in an average item-level content validity index (I-CVI) of 0.89 and a scale-level content validity index (S-CVI/Ave) of 0.92, indicating that the scale accurately reflects the targeted construct at a high level. Moreover, the Cronbach alpha for the cognitive load measurement was 0.813, which implied the high reliability of the measurement used in this study.

RESULTS

The difference of students' thinking skills among the three groups

CT skills

The difference in CT skills under the three programming tools was examined using an analysis of covariance (ANCOVA), with pre-test scores as the covariate and post-test scores as the dependent variable. All test scores were found to be normally distributed based on Kolmogorov–Smirnov tests. Another assumption that has been met is the homogeneity of regression with significant results (F (2, 109) = 3.804, $p > 0.05$). Meanwhile, the F-test results for the product terms of experimental conditions and pre-test CT skills did not violate the homogeneity-of-slopes assumption (F (2, 109) = 0.235, $p > 0.05$), indicating it was sensible to perform the ANCOVA test.

The ANCOVA results showed that the three groups were significantly different in CT skills (F (2, 109) = 4.467, $p < 0.05$, $\eta_p^2 = 0.076$) (as shown in Table 3), with medium effect size (Cohen, 1988). The post hoc results demonstrated that the CT skills of the block programming group were not significantly different from those of the tangible programming group (Sig = 0.397 > 0.05) but were higher than that of the paper-and-pencil programming group (BPG–PPG: $M = 3.451$, $p < 0.05$; TPG–PPG: $M = 2.445$, $p < 0.05$). The students in the block and tangible environments improved their CT skills significantly more than those in the paper-and-pencil environment.

Spatial reasoning skills

ANCOVA, with pre-test scores as covariates and post-test scores as dependent variables, was used to examine the differences in spatial reasoning skills under the three programming tools. The pre-test and post-test scores were found to be normally distributed based on Kolmogorov–Smirnov tests. Another assumption that has been met is the homogeneity of regression with significant results (F (2, 109) = 1.234, $p > 0.05$). Meanwhile, the F-test results for the product terms of experimental conditions and pre-test spatial reasoning skills did not violate the homogeneity-of-slopes assumption (F (2, 109) = 0.540, $p > 0.05$), indicating it was sensible to perform the ANCOVA test.

The ANCOVA results showed that there were significant differences in spatial reasoning skills (F (2, 109) = 42.463, $p < 0.001$, $\eta_p^2 = 0.440$), mental rotation (F (2, 109) = 27.045, $p < 0.001$, $\eta_p^2 = 0.334$), spatial orientation (F (2, 109) = 23.072, $p < 0.001$, $\eta_p^2 = 0.299$) and spatial visualization (F (2, 109) = 14.772, $p < 0.001$, $\eta_p^2 = 0.215$) among the three groups (as shown in Table 4), with a large effect size (Cohen, 1988). The post-hoc results demonstrated that the students in the tangible programming group performed significantly better than the block programming group (TPG–BPG: $M = 2.537$, $p < 0.001$) and paper-and-pencil

TABLE 3 ANCOVA results for CT skills.

Group	N	SD	Adj.M	SE	F	Pairwise comparison
TPG	37	6.669	23.460	0.842	4.467*	TPG > PPG
BPG	38	8.141	24.132	0.830		BPG > PPG
PPG	37	8.326	19.811	0.843		

Note: Group categories: TPG, tangible programming group; BPG, block programming group; PPG, paper-and-pencil programming group.

*$p < 0.05$.

TABLE 4 ANCOVA results for spatial reasoning skills.

Dimension	*N*	*SD*	Adj.*M*	*SE*	*F*	Pairwise comparison
Mental rotation						
TPG	37	1.530	7.950	0.202	27.045**	TPG > BPG > PPG
BPG	38	1.353	6.832	0.199		
PPG	37	1.632	5.844	0.202		
Spatial orientation						
TPG	37	0.702	9.275	0.147	23.072**	TPG > BPG > PPG
BPG	38	0.950	8.728	0.145		
PPG	37	1.167	7.869	0.147		
Spatial visualization						
TPG	37	1.739	8.176	0.210	14.772**	TPG > BPG > PPG
BPG	38	1.541	7.283	0.207		
PPG	37	1.121	6.561	0.210		
Spatial reasoning skills						
TPG	37	3.149	25.367	0.387	42.463**	TPG > BPG > PPG
BPG	38	2.814	22.831	0.382		
PPG	37	2.621	20.320	0.387		

**$p < 0.001$.

programming group (TPG–PPG: $M = 5.047$, $p < 0.001$) in spatial reasoning skills. These results revealed that tangible programming tools positively affected students' spatial reasoning skills more than paper-and-pencil programming tools in GPL.

Performance of GPL task lists

The performance of GPL task lists was analysed in depth to supplement and validate the results of computational thinking and spatial reasoning skills measurements. The results of the one-way ANOVA showed that the three groups differed significantly ($p < 0.05$) in GPL task list results on the five dimensions of abstraction, problem decomposition, algorithm design, algorithm efficiency and spatial reasoning skills. The results are shown in Table 5. Specifically, the tangible programming group scored higher on the abstraction ($M = 4.050$, $SD = 0.790$), problem decomposition ($M = 4.150$, $SD = 0.900$), and spatial reasoning skills ($M = 4.190$, $SD = 0.710$) dimensions. The block programming group has the highest algorithm design ($M = 3.947$, $SD = 0.978$) and algorithm efficiency ($M = 4.290$, $SD = 0.840$). These results of the final GPL task lists validated the results of the CT skills and spatial reasoning skills tests.

The difference of students' learning experience among the three groups

Enjoyment

In order to analyse how different programming tools affect students' enjoyment during GPL, one-way ANOVA was used. The results are shown in Table 6, and the difference in enjoyment between the three groups was not statistically significant (F (2, 109) = 0.020, $p > 0.05$). This result indicated that the students' enjoyment during the GPL process was similar under the three programming tools.

TABLE 5 Descriptive statistics for the GPL task lists' variables.

Variables	Group	*N*	Mean	*SD*	*F*	Pairwise comparison
Abstraction	TPG	37	4.050	0.790	0.170*	TPG > PPG
	BPG	38	3.740	1.020		BPG > PPG
	PPG	37	3.200	0.950		
Problem decomposition	TPG	37	4.150	0.900	0.083*	TPG > PPG
	BPG	38	3.840	1.070		BPG > PPG
	PPG	37	3.220	1.180		
Algorithm design	TPG	37	3.770	0.887	0.046*	BPG > PPG
	BPG	38	3.947	0.978		TPG > PPG
	PPG	37	3.189	1.163		
Algorithm efficiency	TPG	37	3.890	0.880	0.601**	BPG > PPG
	BPG	38	4.290	0.840		TPG > PPG
	PPG	37	3.270	0.930		
Spatial reasoning skills	TPG	37	4.190	0.710	0.393**	TPG > BPG > PPG
	BPG	38	3.720	0.840		
	PPG	37	3.140	0.770		

**$p < 0.001$; *$p < 0.05$.

Cognitive load

To examine whether there was a significant difference in cognitive load among the three groups, a one-way ANOVA was used, and the results are shown in Table 7, with no significant difference in mental load among the three groups (F (2, 109) = 2.307, $p > 0.05$). However, there was a significant difference in mental effort (F (2, 109) = 7.450, $p < 0.05$), as the mental effort was related to the format and manner in which the information was presented, and the difference reflected the different interaction complexity of the three programming tools. Based on Kolmogorov–Smirnov tests, the test scores were determined to follow a normal distribution. And, Tukey's honestly significant difference (HSD) method was used for post hoc tests, and the mental effort of all three groups was homoscedastic (F (2, 109) = 1.183, $p > 0.05$), satisfying Levene's homogeneity variance of assumption. It was found that the paper-and-pencil programming group produced significantly higher mental effort than the tangible programming group (PPG–TPG: $M = 0.865$, $p < 0.05$) and the block programming group (PPG–BPG: $M = 0.767$, $p < 0.05$). There was no statistically significant difference between tangible and block programming tools (Sig = 0.916 > 0.05).

The post-test results showed that the students in the paper-and-pencil programming group ($M = 5.135$, $SD = 0.918$) had significantly higher cognitive load than the tangible programming group ($M = 4.527$, $SD = 0.986$) and block programming group ($M = 4.500$, $SD = 0.973$).

DISCUSSIONS AND IMPLICATIONS

The effect of the three programming tools on students' CT skills in GPL

We found no significant difference in CT skills between the tangible and block programming students, but both groups scored higher than the paper-and-pencil programming group.

TABLE 6 ANOVA results of students' enjoyment in three groups.

Group	*N*	Mean	*SD*	*F*
TPG	37	18.860	3.449	0.020
BPG	38	18.760	3.166	
PPG	37	18.730	2.434	

TABLE 7 ANOVA results of cognitive load in three groups.

Dimension	Group	*N*	Mean	*SD*	*F*	Pairwise comparison
Mental load	TPG	37	4.760	1.038	2.307	
	BPG	38	4.610	1.054		
	PPG	37	5.110	1.022		
Mental effort	TPG	37	4.300	1.024	7.450*	PPG > TPG
	BPG	38	4.390	1.028		PPG > BPG
	PPG	37	5.160	1.118		
Cognitive load	TPG	37	4.527	0.986	5.213*	PPG > TPG
	BPG	38	4.500	0.973		PPG > BPG
	PPG	37	5.135	0.918		

*$p < 0.05$.

Moreover, for the tangible and block programming groups, there were differences in the sub-dimensions CT skills (ie, abstraction, problem decomposition and algorithm skills) between the two groups, which were confirmed by the students' GPL task lists.

The tangible programming group scored the highest on abstraction and problem decomposition. Regarding abstraction, tangible programming tools went beyond the flatness of previous programming tools by presenting code cards as physical objects, giving students more perceptual stimulation and allowing them to simulate abstract code instructions with more direct physical movements. This is consistent with Lawhead et al. (2002), who noted that robotic programming could provide visual and physical interaction to support learning abstraction concepts. Regarding problem decomposition, physical code cards could help students more clearly disaggregate each step in the tangible programming tools environment, such as breaking down the complex problem into minor, easy-to-work problems according to functional elements, roles or other logic, simplifying the programming problem-solving steps.

The block programming group scored highest in algorithm design and algorithm efficiency. Regarding algorithm design, the game platform used by the block programming group supports students in continuously debugging their programs, allowing a closer observation of how the programming logic works (Sırakaya & Alsancak Sırakaya, 2020). In particular, the Vary-One-Thing-At-Time (VOTAT) strategy helped students think and test their programs' soundness step by step, resolving program anomalies and bugs one by one and clarifying ambiguous problems (Greiff et al., 2015). Regarding algorithm efficiency, Lahtinen et al. (2005) showed that the more realistic and concrete the learning situation was, the more likely it was to lead to learning. Block programming tools could demonstrate differences in algorithmic efficiency through 3D game scenarios, such as executing algorithms with fewer blocks that are more efficient.

The effect of the three programming tools on students' spatial reasoning skills in GPL

In terms of spatial reasoning skills, previous studies have demonstrated that programming robot navigation involves behavioural manipulations related to spatial environment and spatial knowledge. In the current study, we found that regardless of which tool they used, students all improved spatial reasoning skills, which indicated that the form of three-view drawing and stereogram transformation were generally effective in helping learners turn plan information into three-dimensional images (Huang & Lin, 2017). Further, the tangible programming group students who used magnetic blocks performed better in the three groups. This result was consistent with the research conclusions of Baykal et al. (2018), which suggested that the tangible user interface (TUI) enhances spatial reasoning skills by incorporating the inherently spatial nature of physicality. Specifically, the current study designed the magnet blocks for students to engage in timely tangible interactions and experience various spatial behavioural manipulations, such as assembling, rotating, moving, positioning and viewing the 'three-dimensional maze' from different perspectives. Learning stimulus rises from the original visual to tactile media. According to the information-processing theory, individuals perceive the world through various sensory stimuli, and a larger diversity in stimuli sources creates more reliable knowledge (Shams & Seitz, 2008). This also supported other research that students often rely on more specific representations to support their spatial reasoning (Moore et al., 2020).

Students in the tangible programming group fared better than those in the other two groups in terms of mental rotation, spatial orientation and spatial visualization. These findings can also be supported by feedback angles. Students in the paper-and-pencil programming group were requested to create a route for the robot to take in order to reach the blue treasure. However, to do so, they first had to conceive the robot's trajectory and were not given instant feedback. On the contrary, the students in the tangible programming group received instant physical output feedback that allowed them to affirm or modify the route based on the robot's position (Misirli et al., 2019), significantly improving spatial orientation and mental rotation. Consistent with previous studies (Antle, 2013), this result supported the idea that robotic activities can improve spatial abilities.

The effect of the three programming tools on students' enjoyment in GPL

We found no significant difference in enjoyment among the three groups of students and all of them scored higher. This is consistent with Xie et al. (2008)'s research conclusion that there is no difference in the enjoyment of three different interface styles (physical, graphical and tangible). Moreover, one possible reason is that the research introduced a gamification strategy that adopted a robot treasure hunt as the narrative context, utilized a competitive mechanic pushing students to complete the barrier task, and evaluated the students' learning outcomes using gamification elements such as points. Werbach and Hunter (2012) indicated that rewards or competition as extrinsic motivation increase momentary positive actions, and intrinsic motivation sustains students' long-term enjoyment. In addition, many studies have also shown that enjoyment was associated with intrinsic motivation (Ryan & Deci, 2017). We built game objectives and competitive mechanisms, which improved the enjoyment.

The effect of the three programming tools on students' cognitive load in GPL

We found that there was no significant difference among the three groups in terms of the mental load, a finding that was consistent with cognitive load theory (Paas & Van Merriënboer, 1994), where the interactions between the gamification programming task and student characteristics were similar among the three groups. Thus, programming tools matching the cognitive level of novices encapsulate programming code into chunks that do not involve complex programming syntax rules, thereby reducing students' perceived programming task difficulty. Furthermore, the three programming tools impacted students' cognitive load during GPL, especially mental effort. The tangible and block programming groups scored significantly lower on mental effort than those in the paper-and-pencil programming group. This finding affirmed that the well-designed code cards did not increase students' cognitive loads and that their interaction complexity and manoeuvrability were effective in teaching programming. They allowed students to interact with the environment with more physical movement, effectively embedding their cognitive activities in the environment and thereby reducing cognitive loads (Pouw et al., 2014). Compared to the first two groups, when treated with the paper-and-pen programming tool, students may feel that there are no actionable objects to help understand programming concepts, and it is difficult to obtain additional feedback from programming tools, which requires higher cognitive abilities and thus increases their mental effort. This finding was echoed by Huang et al. (2023), who suggested that greater complexity could result in higher mental effort.

Implications

Our key finding is that utilizing a tangible programming tool to convert a three-view drawing into a stereogram has the best impact on developing spatial reasoning skills. This is mainly attributable to the tool's interactivity and embodiment. Therefore, we highly recommend adopting tangible programming tools in future programming instruction targeted at enhancing spatial abilities. Additionally, our research unveiled the positive influence of block programming tools on algorithm design and thinking, which encourages instructors to adopt 3D gamification block programming as a useful learning and practical teaching environment. Importantly, our result supports and adds to other studies on the effectiveness of programming tools. We reaffirm the value of gamification as an optimization strategy in programming education by comparing tangible, block and paper-and-pencil programming tools. This reaffirms that the constraints of paper-and-pencil programming tools can be mitigated by incorporating gamification elements and mechanisms to improve the learning experience. In conclusion, paper-and-pencil programming tools are still flexible and valuable without tangible and block programming tools.

CONCLUSIONS, LIMITATIONS AND FUTURE DIRECTION

Programming tools are critical to success in the GPL field, and their different attributes to differences in students' thinking skills and learning experience. In this study, we designed three programming tools (eg, paper-and-pencil, block, tangible) to investigate the effects of different programming tools on students' CT skills, spatial reasoning skills, enjoyment and cognitive load during the GPL process. The current study aimed to enhance CT and spatial reasoning skills, vital for the 21st century. From the empirical results, results showed significant improvements in students' abstract, problem decomposition and spatial reasoning skills with tangible programming tools. Block programming tools notably boosted algorithm design and efficiency. Gamification helped students maintain their enjoyment of GPL. Both tangible and block programming tools reduced cognitive load compared to paper-and-pencil. Choosing the suitable programming tool is essential for developing specific thinking skills or learning experience in GPL education.

The research also had three limitations. First, only pre-tests and post-tests were conducted, however, understanding the dynamic learning trajectories of students is essential to improve the implementation effectiveness of CT skills education. In the future, we will continue to collect process data on students' programming learning and use the evidence-centred design (ECD) model to code and analyse the trajectories of CT-related skills. Second, we conducted a single 150-minute experiment, and in the future, we will explore the possibility of conducting a longitudinal study to investigate the long-term effects of these programming tools on students' CT and spatial reasoning skills. Third, in the current research, we designed physical blocks to build a stereogram, but the physical material takes up more space and is often expensive. Some previous studies highlighted that virtual learning activities (Molina-Carmona et al., 2018) and VR environments (Gün & Atasoy, 2017) could contribute to developing spatial reasoning skills. Therefore, VR technology can be used in the future to replace physical tools.

ACKNOWLEDGEMENTS

We thank Ms. WANG Ping for her experimental support of this article.

FUNDING INFORMATION

This study was funded by the 2022 Key research project of the Chinese Ministry of Education (DCA220449).

CONFLICT OF INTEREST STATEMENT

There is no potential conflict of interest in this study.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author.

ETHICS STATEMENT

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee.

ORCID

Xin Gong https://orcid.org/0000-0001-5962-0448
Weiqi Xu https://orcid.org/0000-0002-9586-0075
Shufan Yu https://orcid.org/0000-0001-6845-0065
Jingjing Ma https://orcid.org/0000-0003-3507-0941
Ailing Qiao https://orcid.org/0009-0003-6996-7711

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*, 105954. https://doi.org/10.1016/j.chb.2019.03.018

Antle, A. N. (2013). Exploring how children use their hands to think: An embodied interactional analysis. *Behaviour & Information Technology*, *32*(9), 938–954. https://doi.org/10.1080/0144929X.2011.630415

Barradas, R., Lencastre, J., Soares, S., & Valente, A. (2020). Developing computational thinking in early ages: A review of the code.Org platform. In *Proceedings of the 12th International Conference on Computer Supported Education* (pp. 157–168). Prague, Czech Republic. https://doi.org/10.5220/0009576801570168

Baykal, G. E., Alaca, I. V., Yantaç, A. E., & Göksun, T. (2018). A review on complementary natures of tangible user interfaces (TUIs) and early spatial learning. *International Journal of Child-Computer Interaction*, *16*, 104–113. https://doi.org/10.1016/j.ijcci.2018.01.003

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

Carreño-León, M., Sandoval-Bringas, A., Álvarez-Rodríguez, F., & Camacho-González, Y. (2018). Gamification technique for teaching programming. In *2018 IEEE Global Engineering Education Conference (EDUCON)* (pp. 2009–2014). Santa Cruz de Tenerife, Spain. https://doi.org/10.1109/EDUCON.2018.8363482

Cheng, Y. P., Lai, C. F., Chen, Y. T., Wang, W. S., Huang, Y. M., & Wu, T. T. (2023). Enhancing student's computational thinking skills with student-generated questions strategy in a game-based learning platform. *Computers & Education*, *200*, 104794. https://doi.org/10.1016/j.compedu.2023.104794

Chittum, J. R., Jones, B. D., Akalin, S., & Schram, Á. B. (2017). The effects of an afterschool STEM program on students' motivation and engagement. *International Journal of STEM Education*, *4*(1), 1–16. https://doi.org/10.1186/s40594-017-0065-4

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

Città, G., Gentile, M., Allegra, M., Arrigo, M., Conti, D., Ottaviano, S., Reale, F., & Sciortino, M. (2019). The effects of mental rotation on computational thinking. *Computers & Education*, *141*, 103613. https://doi.org/10.1016/j.compedu.2019.103613

Clarke-Midura, J., Silvis, D., Shumway, J. F., Lee, V. R., & Kozlowski, J. S. (2021). Developing a kindergarten computational thinking assessment using evidence-centered design: The case of algorithmic thinking. *Computer Science Education*, *31*(2), 117–140. https://doi.org/10.1080/08993408.2021.1877988

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Academic Press.

Dagienė, V., & Sentance, S. (2016). It's computational thinking! Bebras tasks in the curriculum. In A. Brodnik & F. Tort (Eds.), *Informatics in Schools: Improvement of Informatics Knowledge and Perception: 9th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2016, Münster, Germany, October 13–15, 2016, Proceedings 9* (pp. 28–39). Springer International Publishing. https://doi.org/10.1007/978-3-319-46747-4_3

Dilmen, K., Kert, S. B., & Uğraş, T. (2023). Children's coding experiences in a block-based coding environment: A usability study on code.org. *Education and Information Technologies*, *28*, 1–26. https://doi.org/10.1007/s10639-023-11625-8

Dúo-Terrón, P. (2023). Analysis of scratch software in scientific production for 20 years: Programming in education to develop computational thinking and STEAM disciplines. *Education Sciences*, *13*(4), 404. https://doi.org/10.3390/educsci13040404

Erol, O., & Kurt, A. A. (2017). The effects of teaching programming with scratch on pre-service information technology teachers' motivation and achievement. *Computers in Human Behavior*, *77*, 11–18. https://doi.org/10.1016/j.chb.2017.08.017

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016

Figueiredo, J., & Garcia-Penalvo, F. J. (2020). Increasing student motivation in computer programming with gamification. In *2020 IEEE Global Engineering Education Conference (EDUCON)* (pp. 997–1000). Porto, Portugal. https://doi.org/10.1109/EDUCON45650.2020.9125283

Garcia, M. B. (2021). Cooperative learning in computer programming: A quasi-experimental evaluation of Jigsaw teaching strategy with novice programmers. *Education and Information Technologies*, *26*(4), 4839–4856. https://doi.org/10.1007/s10639-021-10502-6

Greiff, S., Wüstenberg, S., & Avvisati, F. (2015). Computer-generated log-file analyses as a window into students' minds? A showcase study based on the PISA 2012 assessment of problem solving. *Computers & Education*, *91*, 92–105. https://doi.org/10.1016/j.compedu.2015.10.018

Gün, E. T., & Atasoy, B. (2017). The effects of augmented reality on elementary school students' spatial ability and academic achievement. *Egitim ve Bilim-Education and Science*, *42*(191), 31–51. https://doi.org/10.15390/EB.2017.7140

Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., & Edwards, T. (2016). Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning. *Computers in Human Behavior*, *54*, 170–179. https://doi.org/10.1016/j.chb.2015.07.045

Hegarty, M., Montello, D. R., Richardson, A. E., Ishikawa, T., & Lovelace, K. (2006). Spatial abilities at different scales: Individual differences in aptitude-test performance and spatial-layout learning. *Intelligence*, *34*(2), 151–176. https://doi.org/10.1016/j.intell.2005.09.005

Hu, Y., Chen, C. H., & Su, C. Y. (2021). Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis. *Journal of Educational Computing Research*, *58*(8), 1467–1493. https://doi.org/10.1177/0735633120945935

Huang, S. Y., Tarng, W., & Ou, K. L. (2023). Effectiveness of AR board game on computational thinking and programming skills for elementary school students. *Systems*, *11*(1), 25. https://doi.org/10.3390/systems11010025

Huang, T. C., & Lin, C. Y. (2017). From 3D modeling to 3D printing: Development of a differentiated spatial ability teaching model. *Telematics and Informatics*, *34*(2), 604–613. https://doi.org/10.1016/j.tele.2016.10.005

Jordan, P. W. (2002). *Designing pleasurable products: An introduction to the new human factors*. Taylor & Francis.

Kim, B., Kim, T., & Kim, J. (2013). Paper-and-pencil programming strategy toward computational thinking for non-majors: Design your solution. *Journal of Educational Computing Research*, *49*(4), 437–459. https://doi.org/10.2190/EC.49.4.b

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14–18. https://doi.org/10.1145/1151954.1067453

Lawhead, P. B., Bland, C. G., Barnes, D. J., Duncan, M. E., Goldweber, M., Hollingsworth, R. G., & Schep, M. (2002). A road map for teaching introductory programming using LEGO© mindstorms robots. *ACM SIGCSE Bulletin*, *35*(2), 191–201. https://doi.org/10.1145/782941.783002

Lee, L. K., Cheung, T. K., Ho, L. T., Yiu, W. H., & Wu, N. I. (2021). A cross-platform game for learning computational thinking with the support of collaborative learning. *International Journal of Innovation and Learning*, *30*(3), 334–357. https://doi.org/10.1504/IJIL.2021.118188

Lindberg, R. S. N., Laine, T. H., & Haaranen, L. (2018). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games. *British Journal of Educational Technology*, *50*(4), 1979–1995. https://doi.org/10.1111/bjet.12685

Liu, F., Zhao, L., Zhao, J., Dai, Q., Fan, C., & Shen, J. (2022). Educational process mining for discovering students' problem-solving ability in computer programming education. *IEEE Transactions on Learning Technologies*, *15*(6), 709–719. https://doi.org/10.1109/TLT.2022.3216276

Melcer, E., & Isbister, K. (2017). Embodiment, collaboration, and challenge in educational programming games: Exploring use of tangibles and mouse. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (pp. 1–6). Hyannis, Massachusetts. https://doi.org/10.1145/3102071.3116222

Misirli, A., Komis, V., & Ravanis, K. (2019). The construction of spatial awareness in early childhood: The effect of an educational scenario-based programming environment. *Review of Science, Mathematic & ICT Education*, *13*(1), 111–124. https://doi.org/10.26220/rev.3122

Molina-Carmona, R., Pertegal-Felices, M., Jimeno-Morenilla, A., & Mora-Mora, H. (2018). Virtual reality learning activities for multimedia students to enhance spatial ability. *Sustainability*, *10*(4), 1074. https://doi.org/10.3390/su10041074

Moore, T. J., Brophy, S. P., Tank, K. M., Lopez, R. D., Johnston, A. C., Hynes, M. M., & Gajdzik, E. (2020). Multiple representations in computational thinking tasks: A clinical study of second-grade students. *Journal of Science Education and Technology*, *29*(1), 19–34. https://doi.org/10.1007/s10956-020-09812-0

Ou-Yang, F. C., Lai, H. M., & Wang, Y. W. (2023). Effect of augmented reality-based virtual educational robotics on programming students' enjoyment of learning, computational thinking skills, and academic achievement. *Computers & Education*, *195*, 104721. https://doi.org/10.1016/j.compedu.2022.104721

Paas, F., Tuovinen, J. E., Tabbers, H., & Van Gerven, P. W. M. (2003). Cognitive load measurement as a means to advance cognitive load theory. *Educational Psychologist*, *38*(1), 63–71. https://doi.org/10.1207/S15326985EP3801_8

Paas, F. G. W. C., & Van Merriënboer, J. J. G. (1994). Instructional control of cognitive load in the training of complex cognitive tasks. *Educational Psychology Review*, *6*(4), 351–371. https://doi.org/10.1007/BF02213420

Pearce, J., Ainley, M., & Howard, S. (2005). The ebb and flow of online learning. *Computers in Human Behavior*, *21*(5), 745–771. https://doi.org/10.1016/S0747-5632(04)00036-6

Pérez-Marín, D., Hijón-Neira, R., & Pizarro, C. (2022). Coding in early years education: Which factors influence the skills of sequencing and plotting a route, and to what extent? *International Journal of Early Years Education*, *30*(4), 969–985. https://doi.org/10.1080/09669760.2022.2037076

Pouw, W. T. J. L., Van Gog, T., & Paas, F. (2014). An embedded and embodied cognition review of instructional manipulatives. *Educational Psychology Review*, *26*(1), 51–72. https://doi.org/10.1007/s10648-014-9255-5

Quevedo Gutierrez, E., & Zapatera Llinares, A. (2021). Assessment of scratch programming language as a didactic tool to teach functions. *Education Sciences*, *11*(9), 499. https://doi.org/10.3390/educsci11090499

Ramful, A., Lowrie, T., & Logan, T. (2016). Measurement of spatial ability: Construction and validation of the spatial reasoning instrument for middle school students. *Journal of Psychoeducational Assessment*, *35*(7), 1–19. https://doi.org/10.1177/0734282916659207

Robert, L., Miyahara, D., Lafourcade, P., Libralesso, L., & Mizuki, T. (2022). Physical zero-knowledge proof and NP-completeness proof of Suguru puzzle. *Information and Computation*, *285*, 104858. https://doi.org/10.1016/j.ic.2021.104858

Rojas-López, A., Rincón-Flores, E. G., Mena, J., García-Peñalvo, F. J., & Ramírez-Montoya, M. S. (2019). Engagement in the course of programming in higher education through the use of gamification. *Universal Access in the Information Society*, *18*(3), 583–597. https://doi.org/10.1007/s10209-019-00680-z

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, *72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Ryan, R. M., & Deci, E. L. (2017). *Self-determination theory. Basic psychological needs in motivation, development and wellness*. The Guilford Press.

Sapounidis, T., & Demetriadis, S. (2013). Tangible versus graphical user interfaces for robot programming: Exploring cross-age children's preferences. *Personal and Ubiquitous Computing*, *17*(8), 1775–1786. https://doi.org/10.1007/s00779-013-0641-7

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

Shaban, A., Pearson, E., & Chang, V. (2021). Evaluation of user experience, cognitive load, and training performance of a gamified cognitive training application for children with learning disabilities. *Frontiers in Computer Science*, *3*, 617056. https://doi.org/10.3389/fcomp.2021.617056

Shams, L., & Seitz, A. R. (2008). Benefits of multisensory learning. *Trends in Cognitive Sciences*, *12*(11), 411–417. https://doi.org/10.1016/j.tics.2008.07.006

Sharma, K., Papavlasopoulou, S., & Giannakos, M. (2019). Coding games and robots to enhance computational thinking: How collaboration and engagement moderate children's attitudes? *International Journal of Child-Computer Interaction*, *21*, 65–76. https://doi.org/10.1016/j.ijcci.2019.04.004

Sırakaya, M., & Alsancak Sırakaya, D. (2020). Augmented reality in STEM education: A systematic review. *Interactive Learning Environments*, *30*(8), 1556–1569. https://doi.org/10.1080/10494820.2020.1722713

Song, J. B. (2019). The effectiveness of an unplugged coding education system that enables coding education without computers. *Universal Journal of Educational Research*, *10*(1), 129–137. https://doi.org/10.13189/ujer.2019.071514

Souza, A. C. D., Alexandre, N. M. C., Guirardello, E. D. B., Souza, A. C. D., Alexandre, N. M. C., & Guirardello, E. D. B. (2017). Propriedades psicométricas na avaliação de instrumentos: Avaliação da confiabilidade e da validade. *Epidemiologia e Serviços de Saúde*, *26*(3), 649–659. https://doi.org/10.5123/S1679-49742017000300022

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education*, *25*(3), 293–319. https://doi.org/10.1007/s10798-014-9287-7

Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teachers' rules: Exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*, *28*(2), 347–376. https://doi.org/10.1007/s10798-017-9400-9

Sun, L., Hu, L., & Zhou, D. (2021). Which way of design programming activities is more effective to promote K-12 students' computational thinking skills? A meta-analysis. *Journal of Computer Assisted Learning*, *37*(4), 1048–1062. https://doi.org/10.1111/jcal.12545

Sweller, J., van Merrienboer, J. J. G., & Paas, F. G. W. C. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, *10*(3), 251–296. https://doi.org/10.1023/A:1022193728205

Wang, X. Y., Xing, Q., Jin, Q., & Wang, D. L. (2024). "Be a lighting programmer": Supporting children collaborative learning through tangible programming system. *International Journal of Human–Computer Interaction*, *40*(10), 2622–2640. https://doi.org/10.1080/10447318.2022.2163783

Wang, Y. (2023). Can gamification assist learning? A study to design and explore the uses of educational music games for adults and young learners. *Journal of Educational Computing Research*, *60*(8), 2015–2035. https://doi.org/10.1177/07356331221098148

Werbach, K., & Hunter, D. (2012). *For the win: How game thinking can revolutionize your business*. Wharton Digital Press.

Xie, L., Antle, A. N., & Motamedi, N. (2008). Are tangibles more fun?: Comparing children's enjoyment and engagement using physical, graphical and tangible user interfaces. In *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction* (pp. 191–198). Bonn, Germany. https://doi.org/10.1145/1347390.1347433

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

Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S., & Lan, X. (2022). The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers and Education: Artificial Intelligence*, *3*, 100096. https://doi.org/10.1016/j.caeai.2022.100096

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

How to cite this article: Gong, X., Xu, W., Yu, S., Ma, J., & Qiao, A. (2024). Enhancing computational thinking and spatial reasoning skills in gamification programming learning: A comparative study of tangible, block and paper-and-pencil tools. *British Journal of Educational Technology*, *00*, 1–23. https://doi.org/10.1111/bjet.13482