education sciences

MDPI

Article

Cultivating Creativity and Improving Coding Skills in Primary School Students via Domain-General and Domain-Specific Learning Scaffoldings

Shih-Wen Su *, Li-Xian Chen, Shyan-Ming Yuan and Chuen-Tsai Sun

Department of Computer Science, Institute of Computer Science and Engineering, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan; lixian.cs98g@g2.nctu.edu.tw (L.-X.C.); smyuan@nycu.edu.tw (S.-M.Y.); ctsun@cs.nycu.edu.tw (C.-T.S.)

* Correspondence: alvin.cs00@nycu.edu.tw

Abstract: The transformative wave of generative AI is reshaping the creative thinking processes of learners, posing a significant challenge to education and industry in cultivating technological literacy and creativity. This study delves into the exploration of how learners can effectively tackle new challenges by deconstructing fragments from a macro perspective and generating innovative methods or concepts. In the Scratch visual programming environment, learners in the self-regulated learning mode observed entire functioning projects, facilitating easy disassembly and learning, namely by using Code Decomposed by Learner (CDBL). A total of 104 fifth-grade students were divided into two learning scaffoldings: (1) domain-general, learning from the top down (CDBL-TD), and (2) domain-specific, learning from the bottom up (CDBL-BU). Students in the CDBL-TD group exhibited a high degree of completion, strong exploration abilities, and the willingness to experiment with unlearned functions. Although there was no significant difference in originality between the two groups, students in the CDBL-TD group showcased greater uniqueness in designing characters or items within the game. This study introduces a novel programming learning scaffolding, offering instructors a tool to guide students' creativity and enhance their programming capabilities.

Keywords: self-regulated learning; visual programming; creativity; coding; Scratch; learning scaffolding methods; primary education

Citation: Su, S.-W.; Chen, L.-X.; Yuan, S.-M.; Sun, C.-T. Cultivating Creativity and Improving Coding Skills in Primary School Students via Domain-General and Domain-Specific Learning Scaffoldings. *Educ. Sci.* **2024**, *14*, 695. https://doi.org/10.3390/educsci14070695

Academic Editor: Fred Paas

Received: 23 May 2024
Revised: 23 June 2024
Accepted: 24 June 2024
Published: 26 June 2024

1. Introduction

In recent years, to foster students' hands-on problem-solving abilities, educational institutions at various levels have extensively undertaken numerous investigations into STEM education [1]. Computer science, specifically proficiency in utilizing computers for mathematical problem solving, has been recognized as a fundamental STEM subject for all K-12 students [2,3]. Researchers have concentrated on cultivating the aptitude of students in robotics and other domains that demand computer science and engineering skills [3].

However, the use of traditional programming languages in education, with their strict syntax rules, has led to a loss of interest in programming among many students [4–8]. However, traditional programming language teaching methods have led to a decline in student interest, attributable to the rigid grammatical rules and fragmented approaches to grammar instruction. This issue has been addressed through the introduction of visual programming software. Innovative approaches to computer science programming education, integrating visual programming into primary school curricula, have been implemented in various countries, including the United States, the United Kingdom, Japan, and Taiwan [9–11].

Visual programming, manifested in block-based coding, has played a pivotal role in computer science education. This approach enables students to concentrate on programming concepts and structures rather than programming language operations [12,13]. It has

effectively bolstered student confidence in learning programming by alleviating the cognitive load and reducing learning anxiety [14–17] Well-known visual programming tools such as Scratch [5,18,19], Alice [20,21], Code.org [16,22], and MIT App Inventor [23,24] have gained widespread usage in primary and secondary school computer courses. Taiwan's Ministry of Education has used Scratch to hold national Scratch competitions from 2014 to the present. This tool introduces a novel approach to self-regulated learning in STEM education, fostering increased creativity in developing games. Therefore, this study aims to explore the effectiveness of the tool in cultivating primary school students' programming skills and creativity.

Hands-on, inquiry-based learning establishes an environment wherein students actively engage in problem discovery and solution-finding under the guidance of teachers, aiming to cultivate their problem-solving abilities [25]. Nobel Prize winner in physics Georges Charpak and French Academy of Sciences academician Yves Quéré advocate for a "learning by doing" teaching model, enabling learners to grasp basic concepts from the top-down to the bottom-up level [26]. In contrast to traditional programming teaching, which typically starts from the bottom and introduces concepts like loops, functions, arrays, indicators, and strings in separate chapters, this bottom-up approach can pose challenges. When students attempt coding projects, they often face hurdles such as blocked flows, unclear input and output, and errors in data type conversion. We have observed similar issues among novice programmers at the university level [27]. This has prompted us to investigate whether similar problems arise in other stages of learning. In the current era of rapid big data and artificial intelligence development, there is a growing emphasis on promoting students' logical thinking and programming abilities as crucial skills for the future workplace. Effectively transforming the traditional computer teaching model to provide students with diverse opportunities to learn fundamental knowledge, skills, and problem-solving attitudes in programming is a vital concern within the education community and among scholars.

This study aims to explore innovative methods for learning programming (CDBL-TD and CDBL-BU) to foster primary school students' programming skills and creativity. The building-block-style visual programming language, Scratch, is used to compare changes in students' programming skills and creativity after experiencing different programming teaching methods. Through learning activities in a semester's computer class, we aim to find a more efficient way of developing programming skills and logical thinking. This approach equips students with the skills required to solve real-life problems and systematically improves their competitiveness.

1.1. Self-Regulated Learning in Reverse Engineering

The reverse engineering teaching method in science education requires students to reconstruct or optimize complete products to achieve micro-innovation results [28]. Reverse engineering was initially aimed at analyzing the interrelationships of components in existing engineering systems or based on the use of existing products to restore prototypes and develop new products [29–31]. The teaching method offers a meaningful purpose for learning, rendering reverse engineering an effective educational tool [32–35]. This study, leveraging the reversible assembly characteristics of Scratch, adapted the reverse engineering learning method. This process contributes to enhancing students' motivation, critical thinking [36,37], logical thinking [38], and understanding of current products [39,40]. The self-regulated learning method underscores the significance of STEM education. However, students often need more knowledge about the engineering methods used to create or complete the product. The challenge lies in acquiring a working knowledge of the original design by disassembling the product. Therefore, during the program deconstruction process, students continually self-regulate to form relevant working knowledge, enabling them to complete new program projects.

Self-regulated learning emphasizes students' independence in learning, encompassing their ability to self-control, self-supervise, and learn autonomously [41–44]. Students

self-regulate in order to organize study time effectively, resulting in improved learning performance [45]. Additionally, the learning scaffolding theory also supports the development of self-regulated learning [46,47]. Integrating self-regulated learning into program deconstruction can significantly enhance the efficiency of programming learning. This addresses the traditional challenge of limited teaching hours, where students are constrained to follow the class's pace, leading to a lack of critical thinking and a tendency to replicate teacher-taught programs without a deep understanding of their significance, eventually diminishing interest in programming [4–8]. In the autonomous learning and deconstruction process, students attain learning autonomy by controlling the learning pace through self-reflection and hands-on processes. The effectiveness of allowing students to demonstrate decomposition and assembly through a self-regulated learning process in university novice programming has been validated [27]. However, a need for more research for novice programmers at the primary school level remains. This study aims to investigate this research gap.

1.2. Stimulating Creativity in Visual Programming Environments

Creativity is a crucial component of STEM education, and scholars have pointed out the application of new technologies to cultivate and enhance learners' creative processes, thereby augmenting the potential for transformative learning [48,49]. Over the past few decades, numerous attempts have been made to teach and inspire creativity in students of all ages. Traditional teaching methods have often failed to achieve these objectives. For instance, Chen and Chiu [50] employed inter-group competition to stimulate students' academic performance and creativity, observing that students increasingly rely on the Internet for information, distancing themselves from the control of classroom or online teachers. Contemporary classroom teaching has evolved from the mere transfer of teachers' knowledge to supporting students' active exploration in acquiring knowledge and skills. This shift aligns with Piaget's [51] insight that students must engage in activities to grasp the intricacies of a problem. Computer technology has successfully demonstrated innovative teaching methods in physics and mathematics education, such as game design and blog publishing [52,53].

Some research is focused on the benefit of game-based learning: digital games allow students to learn autonomously, fostering creative thinking through game design and development [54,55]. Designing mathematics games, in particular, proves instrumental in helping learners to master skills while promoting motivation and a constructive learning philosophy [56–58]. Researchers have increasingly focused on creating games and coding projects through hands-on activities to enhance learning outcomes [59–61]. In game creation, students combine their prior knowledge and skills to develop strategies and create novel and unique objects [62]. Our study leverages the appeal of game creation to novice programmers, enhancing their original application abilities through game creation by using a visual programming environment. In traditional programming courses, students often replicate the steps demonstrated by the teacher, creating similar program outputs. It is challenging to determine whether students genuinely comprehend the material or are merely reproducing the instructional steps to complete the program. From the game design perspective, our research has established a learning environment that facilitates students' learning and experience of programming coding [52]. The "Code Decomposed by Learner" (CDBL) concept allows students to explore the source code of programs in ways that differ from the teacher's instructions, and the CDBL learning model is integrated into the Scratch programming learning process. In the CDBL learning model, students can deconstruct and reconstruct Scratch programming code provided by the teacher, understanding how the program functions through a constructive learning perspective akin to reverse engineering [63]. The deconstruction process enables students to delve into the intricacies of a product, leveraging their acquired knowledge to construct their projects [64]. Furthermore, within the CDBL learning model, students may enhance their understanding of programming through gaming experience, going beyond sole reliance on teacher in-

structions [27]. Combining self-regulated learning principles [42] and reverse engineering methodologies [64,65], we have integrated these concepts to design learning scaffoldings for fifth-grade primary programming novices.

1.3. Learning Scaffolding and Teaching Methods

Within the self-regulated learning framework, students assume control over their learning process, planning their study time and advancing at their own pace, independent of the overall pace of class instruction. This learner-centric approach underscores the significance of learning guidance, such as utilizing instructional flowcharts to support self-regulated learning [66]. However, research on self-regulated learning in programming education still needs to be improved, with most programming education still concentrating on syntax learning [67]. Studies on autonomous learning for beginners in programming are even scarcer. Our research employs the CDBL learning model, offering learning scaffoldings for programming novices to deconstruct game programming blocks.

Learning scaffoldings are broadly categorized into two types: domain-general and domain-specific. Domain-general scaffoldings equip students with concepts and strategies that can be applied across various domains, encompassing skills like problem solving [68]. These scaffoldings aid students in identifying problem-solving strategies, such as locating relevant information to understand a problem and recognizing the importance of using evidence to support arguments. They contribute to students' comprehension of the overall thought process of problem solving and support them in planning, monitoring, and evaluating, irrespective of the content area. Within the framework of domain-general scaffolding, we worked based on the CDBL learning model to design the scaffolding, a top-down form (CDBL-TD); presenting the entire code provided the complete source code for the CDBL-TD group to deconstruct for learning.

In contrast, domain-specific scaffoldings offer novice learners cues regarding which specific content knowledge to employ during problem solving [68,69]. Research on domain-specific scaffolding suggests that a greater amount of excellent domain-specific content knowledge (expertise) correlates with enhanced problem-solving ability in respective domains [70]. Students' familiarity with the background of a problem may enhance their problem-solving ability [71]. Within the framework of domain-specific scaffolding, we worked based on the CDBL learning model to design the scaffolding: bottom-up (CDBL-BU). We provided a complete source code, separated into three specific functional codes (firing bullets, monsters being eliminated, the effects of background and sound) for the CDBL-BU group to deconstruct for learning purposes.

1.4. Purpose of the Study and Research Questions

In order to solve the dilemma in traditional programming teaching, students imitate the teacher's program code to complete the homework without learning, and the student's creativity is difficult to stimulate. This study designed a set of self-regulated learning scaffolding for primary school-level programming beginners, hoping to find a suitable way for students to cultivate CT skills and creativity and contribute to programming education in primary school.

The aim of this study was to explore the impact of CDBL-TD and CDBL-BU learning scaffolding prompts on programming skills and creativity in fifth-grade school students. Drawing from the theoretical foundations and insights provided by the existing literature, three research questions were formulated:

RQ1: Which scaffolding yields better programming learning outcomes for beginners?
RQ2: Which scaffolding encourages students to independently explore untaught functions of Scratch, enhancing their conceptualization and game design?
RQ3: Which scaffolding inspires students to generate more exceptional and original ideas?

2. Method

2.1. Study Design and Participants

Self-regulated learning emphasizes that students can learn independently and adjust their learning through self-control and self-supervision to achieve better learning results [41–44]. Thus, at this stage, students in both groups need to conduct active learning and disassemble the source code of the provided example games on their own without teaching from the teacher. Moreover, this study uses the visual programming language "Scratch" to reduce elementary school program beginners' cognitive load and learning anxiety when learning programming [14–17].

This study proposed a 12-week game design course with one 40 min class per week. The first to fifth weeks (the first phase) were dedicated to familiarizing students with the basic operations of the Scratch program. During this period, all students followed the teacher's instructional steps for learning the program interface and created their main game. Subsequently, in the sixth to twelfth weeks of the course (the second phase), students advanced to creating an advanced shooting game with additional features, building upon the skills acquired from the initial main game. The skills learned during the disassembly process were then used to develop and complete games of the student's design. The final evaluation of the game works created by the two groups, CDBL-TD and CDBL-BU, utilizing distinct program teaching methods, was carried out by experts. In the initial main game for the first phase, the cat must go to the castle without touching three monsters that move left to right at different speeds. In the initial phase, students must complete the game under the teacher's guidance (Figure 1a). In the advanced shooting game for the second phase, the students are asked to add three functions (such as the cat can fire a ball, the ball can destroy the monsters, and the effects of music and background) based on the initial main game. Students need to perform self-regulated learning to disassemble the sample game code provided. They must change the game completed in the initial stage to a shooting-type game screen (Figure 1b).

(**a**) Initial main game. (**b**) Advanced shooting game.

Figure 1. Screenshots of the game screen students need to complete in the initial and second phases.

The participants in this quasi-experimental study comprised 104 fifth-grade students (53 males and 51 females) selected from four randomly and regularly divided classes in Keelung, north Taiwan. The gender distribution in the sample closely mirrored that of all Taiwanese fifth graders in 2023 (51.8% male, 48.2% female). All participants were novices in Scratch and possessed a standard level of computer literacy, including using word processing and drawing software and submitting assignments using cloud drives and FTP software. To ensure instructional consistency and maintain quality, a single teacher with ten years of expertise in information education facilitated all the sessions. Before the

study commenced, the research team conducted a two-hour training workshop for this educator. This training encompassed experimental protocols, instructional content, and effective techniques to guide students in their self-regulated learning journey. The study's implementation took place over a semester, and parental consent was obtained for every participating student.

2.2. Proposed Top-Down and Bottom-Up Learning Scaffolding Teaching Methods

This study introduced the CDBL programmed learning model, incorporating the process of students' self-regulated learning and introducing the CDBL-TD and CDBL-BU program-learning scaffolding methods. The primary objective is to determine which strategy—top-down or bottom-up—is more effective in enhancing novice programmers' programming skills and creative application capabilities within the CDBL learning scaffolding framework.

During the initial phase (such as Table 1), spanning weeks 1–5, all participants underwent training to design the initial main game. They familiarized themselves with programming functions (such as events, motion, looks, and if/else condition) by creating level-breaking games involving tasks such as using the keyboard's arrow keys to control the game character and reach the castle at the top of the screen. Students also learned to set various conditions in the Scratch program, including avoiding contact with three monsters moving at different speeds. Touching any monster resulted in the game ending, while reaching the castle signified the completion of the level. This primary game formed the basis for our pre-test data analysis, investigating potential differences in students' basic programming abilities.

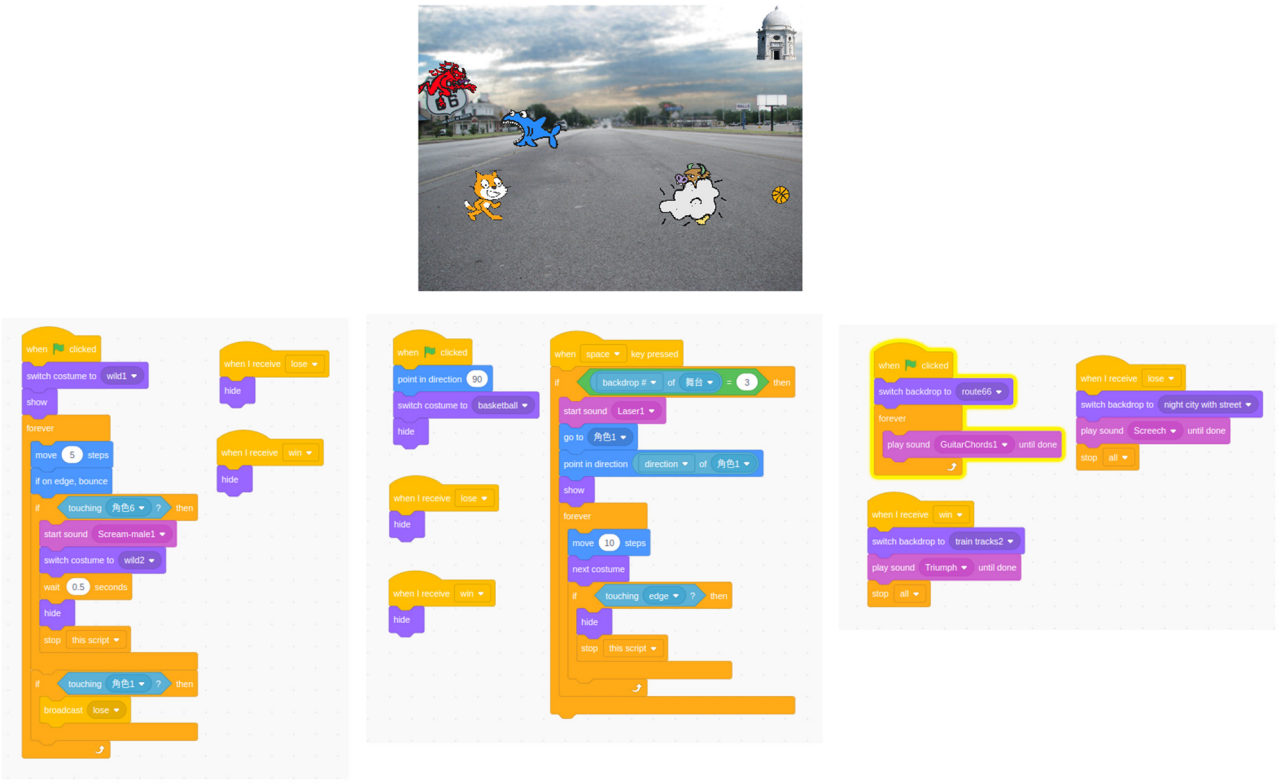
Table 1. The framework of the proposed method.

Week	Requirement	Scratch Functions to Learn	Notation
Initial Phase: Complete Initial Main Game			
1	Control the main character to pass through the castle with arrow keys.	Events, Motion, Looks, If/Else Condition	Teacher teaching
2	Control monsters to move and set failure if the player collides with them.	Loop, If/Else Condition	Teacher teaching
3	Set game level, success, and failure Backgrounds.	Events, If/Else Condition, Sensing Events	Teacher teaching
4	Set sound effects for game clearance, clearance, and failure and display characters.	Sound, Detecting Events, If/Else Condition	Teacher teaching
5	Complete the initial main game.	Debugging skills	Pre-test (completeness, exploratory, originality)
Second Phase: Complete Advanced Shooting Game			
6–11	Students deconstruct the source code to learn the coding skill by themselves. CDBL-TD: Students deconstruct the complete code of the advanced shooting game. CDBL-BU: Students deconstruct three sub-features of the advanced shooting game.	Students explore the new Scratch functions by themselves	Teacher coaching
12	Complete Advanced shooting game.	Debugging skills	Post-test (completeness, exploratory, originality)

In the second phase (such as Table 1), spanning weeks 6 to 12, the methodology aimed to explore students' program-related creative capabilities by integrating a self-regulated learning approach with the CDBL model. During this phase, students were divided

into CDBL-TD and CDBL-BU groups, which required them to dissect the program code independently and explore the new Scratch functions and debugging skills.

In the CDBL-TD group, the students employed an organized and systematic teaching method to help students grasp the main concepts before delving into details. This group adopted a CDBL-TD learning scaffolding approach to deconstruct complete code fragments and learn from them [68]. The 53 students in this group were provided with a complete shooter code. The game screen is shown at the top of Figure 2. Figure 2a shows the program block for the cat firing bullets, Figure 2b shows the program block for the effect of the monster being knocked down, and Figure 2c shows programming blocks for background replacement effects and background music. Student were tasked with using the code's shooting functionality to enhance the original Phase 1 game. They were expected to self-analyze and self-decompose the code, designing a game rooted in their creativity and understanding.

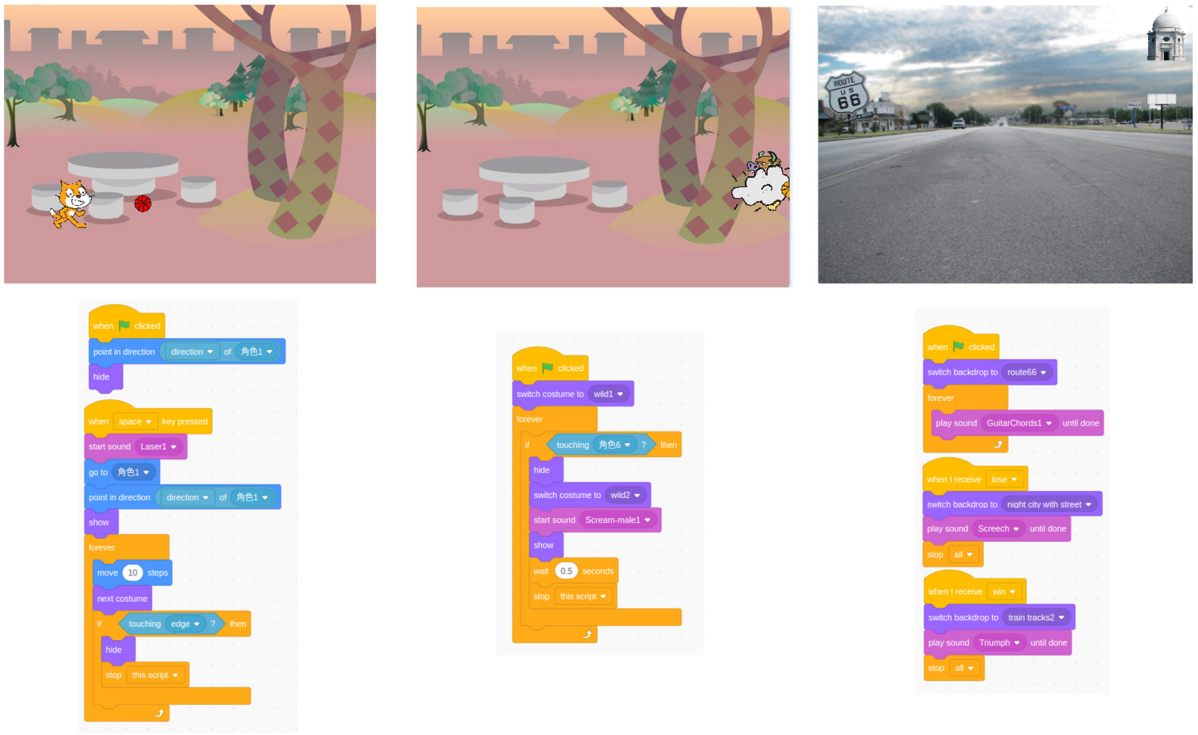


(**a**) The cat can fire a ball. (**b**) The ball can destroy the monster. (**c**) The music and background.

Figure 2. The entire game was provided for the CDBL-TD group.

Conversely, the CDBL-BU group approach emphasized inquiry and exploration, allowing students to consolidate insights as they naturally developed. This group utilized CDBL-BU learning scaffolding to deconstruct and learn from vital functional subroutines separated from the complete code [68,69]. In this group, the teacher demonstrated three functions of a shooting game to 51 students: the character's bullet firing, monster destruction effects, and the combination of background music and sound effects. Figure 3a shows the program blocks and game screen of the cat firing bullets in a separate program, and Figure 3b shows the program blocks for the monster being knocked down in a separate program. Figure 3c shows the program building blocks for the background replacement effect and background music in a separate program. Students were tasked with breaking down these three sub-features of the shooter game and incorporating them into a new shooter game they created. Experts evaluated the programming creativity of the two groups

of students based on the shooting games developed in the second stage. The programming creativity performance served as post-test data for further analysis.



(**a**) The cat can fire a ball. (**b**) The ball can destroy the monster. (**c**) The music and background.

Figure 3. Three independent programs were provided for the CDBL-BU group.

2.3. Programming Skills and Creativity Measurements

This study utilized pre-tests and post-tests to assess students' creativity, focusing on the completeness, exploratory capability, and originality performance of the Scratch code of the games, which were created by students. The scoring process employed the Consensual Assessment Technique (CAT) developed by Amabile [72]. Two expert evaluators were enlisted: an elementary school teacher with fifteen years of experience teaching information and communication technologies and a Ph.D. student in computer science. To avoid cross-influences, these experts independently evaluated the students' games and codes across nine aspects of three dimensions—completeness, exploratory, and originality. Each aspect was assessed using a five-point Likert scale, ranging from "poor" (one point) to "excellent" (five points) [73]. Thus, the highest score of each dimension is fifteen points. In addition, the relation of CDBL-TD and CDBL-BU to these nine aspects of three dimensions is shown in Figure 4. The assessment method for these indicators is outlined below.

For the completeness metric, the evaluation focused on the student's ability to achieve project goals by decomposing code blocks independently. The assessment of game programming results considered students' programming skills based on the correctness of game execution and the completeness of the code. Three major aspects were considered, as defined by Rijo-García et al. [74]:

1. Completion degree, which shows the extent to which the student's game program has been thoroughly designed and completed. It evaluates whether the project goals have been fully realized and implemented.
2. Simplicity, which assesses the effectiveness of the code written by students, ensuring that it accomplishes the task without unnecessary redundancies. It focuses on the clarity and simplicity of the code structure.

3. Correctness, which measures how correct and error-free the student's code is, ensuring that the desired outcome is achieved. It evaluates the accuracy of the code in executing the intended functions without errors.

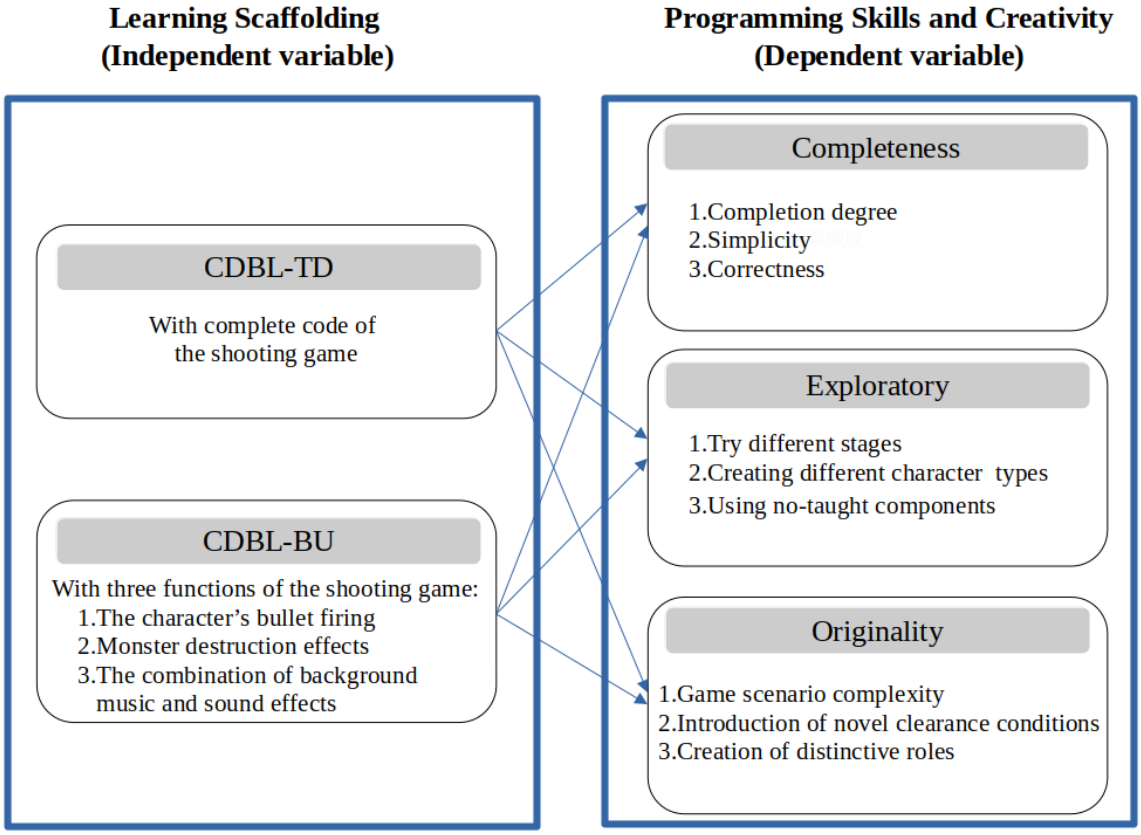


Figure 4. The relation of CDBL-TD and CDBL-BU to completeness, exploratory, and originality.

For the exploratory metric, the evaluation focused on students' ability to incorporate and integrate new elements into their game designs, indicating their enthusiasm to explore beyond the original game's designs. Guilford [75] highlighted that creativity involves generating ideas that differ from an original work or creating more novel objects. In the exploratory dimension, experts assessed whether students modified the original assignment by changing the visuals of the main character, adjusting monster paths, or utilizing untaught Scratch components. This aspect is crucial in determining whether students can flexibly use functions that are not explicitly taught by teachers. The analysis directly examined the game codes created by students to identify self-learned functions that were not part of the taught curriculum. Three aspects were considered:

1. Different stages represent students' ability to design more levels or increase the game's difficulty. This is used to assess whether students expanded the scope of their games beyond the original assignment.
2. Creating different character types means students can create various characters beyond the two game characters included initially. This behavior is used to evaluate their creativity in character design.
3. Using untaught components means that students use program functions or modules the teacher had not taught in the previous two teaching stages. The behavior assesses the extent to which students independently explored and incorporated untaught elements into their game designs.

Originality, conceptualized as the knack for innovating and presenting unforeseen ideas, is scrutinized within the students' game creations. Experts rate originality by considering three aspects:

1. Game scenario complexity refers to the number of elements added to a scene. The more objects present, the higher the score.
2. Introduction of novel clearance conditions, which pertain to the distinctive game-winning criteria introduced by students during the game design process and their level of appropriateness.
3. Create distinctive roles, which focus on whether students have designed unique appearances or movements for the main characters in their games.

A high score is awarded for adding intricate game modes, like unique barriers or monsters, or for devising new game clearance benchmarks, such as procuring additional keys. Unlike exploratory, originality focuses on how students apply their ideas in games. Integrating roles with unique attributes, like a repeatedly throwable ax or a fireball-spewing monster, boosts the originality score.

3. Results

Our first task was to analyze the effects of the two teaching methods on the Scratch project design post-test scores. Descriptive statistics for those different scoring dimensions according to different teaching methods are shown in Table 2. The Scratch project design post-test scores of the CDBL-TD group were completeness (M = 11.89, SD = 2.36), exploratory (M = 9.87, SD = 2.39), and originality (M = 5.60, SD = 1.81). The CDBL-BU group's project design scores included completeness (M = 24.55, SD = 5.82), exploratory (M = 10.75, SD = 2.02), and originality (M = 5.59, SD = 2.56). Overall, the total post-test scores of the CDBL-TD group (M = 27.36, SD = 4.88) were higher than those of the CDBL-BU group (M = 24.55, SD = 5.82).

Table 2. Descriptive statistics for Scratch project design post-test scores for CDBL-TD and CDBL-BU groups.

	Group			
Score	**CDBL-TD (N = 53)**		**CDBL-BU (N = 51)**	
	M	SD	M	SD
Total	27.36	4.88	24.55	5.82
Completeness	11.89	2.36	10.75	2.02
Exploratory	9.87	2.39	8.22	2.45
Originality	5.60	1.81	5.59	2.56

3.1. Effects of Two Code-Teaching Methods on Scratch Project Design Scores

One-way ANCOVA was conducted to determine whether CDBL-TD and CDBL-BU groups' project design post-test scores have statistical significance or not. First, we check whether there is a significant difference between CDBL-TD and CDBL-BU total post-test scores. The within-group homogeneity of regression has no significance between groups and total pre-test scores ($F(1, 101) = 3.049$, $p = 0.084 > 0.05$). Therefore, this study excluded the influence of the total pre-test score and tested whether there was a significant difference in the total post-test scores of CDBL-TD and CDBL-BU students after 12 weeks of Scratch training courses. Leneve's test indicated that the error variance of the two groups of dependent variables was not significantly different and was homogeneous ($F = 0.058$, $p = 0.811 > 0.05$). Results from a one-way ANCOVA indicate a significant difference between two groups' post-test scores ($F(1, 101) = 10.247$, $p = 0.002 < 0.05$, $\eta 2 = 0.092$). The CDBL-TD adopted group mean (M = 27.560, SD = 0.702) is significantly higher than the CDBL-BU adopted group mean (M = 24.340, SD = 0.716). This result showed that the performance of the CDBL-TD total score is better than the CDBL-BU group (Table 3).

Considering the overarching importance of diving deep into individual skill dimensions, the study also executed one-way ANCOVA analyses on the dimensions of completeness, exploratory, and originality, thereby facilitating a more precise interpretation of the

actual impact of each teaching method on the critical skills involved in fostering creativity. Here is the analysis for the post-test scores of the three evaluated abilities.

Table 3. ANCOVA results for Scratch project design post-test scores according to pre-test scores.

	Source of Variance	SS	Df	MS	F	η2
Completeness	group	64.749	1	64.749	16.169 ***	0.138
	pre-test	88.555	1	88.555	22.114 ***	0.180
	residual	404.452	101	4.004		
Exploratory	group	62.068	1	62.068	12.684 **	0.112
	pre-test	102.476	1	102.476	20.942 ***	0.172
	residual	494.227	101	4.893		
Originality	group	0.047	1	0.047	.010	0.052
	pre-test	26.163	1	26.163	5.588 ***	0.000
	residual	472.869	101	4.682		
Total Scores	group	259.292	1	259.292	9.912 ***	0.089
	pre-test	290.708	1	290.708	11.113 ***	0.099
	residual	2642.108	101	26.159		

** $p < 0.01$, *** $p < 0.001$.

Next, we wanted to know whether the post-test scores for the three dimensions (i.e., completeness, exploratory, and originality) varied after different Scratch pedagogies. The within-group homogeneity of regression has no significance between groups and pre-test completeness scores (F (1, 101) = 0.880, $p = 0.350 > 0.05$). Therefore, this study excluded the influence of the completeness pre-test score and tested whether there was a significant difference in the completeness post-test scores of CDBL-TD and CDBL-BU students after Scratch training. Leneve's test indicated that the error variance of the two groups of dependent variables was not significantly different and was homogeneous (F = 2.102, $p = 0.150 > 0.05$). Results from one-way ANCOVA indicate a significant difference between two groups' post-test scores (F (1, 101) = 16.169, $p = 0.000 < 0.05$, η2 = 0.138). The CDBL-TD adopted group mean (M = 12.128, SD = 0.280) is significantly higher than the CDBL-BU adopted group mean (M = 10.495, SD = 0.285). This result showed that, by using different Scratch teaching to design a new Scratch code project, the CDBL-TD completeness score is improved to be better than that of the CDBL-BU group (Table 3).

Then, we wanted to know whether statistical significance existed between groups and pre-test exploratory scores. The within-group homogeneity of regression showed no significance between groups and pre-test exploratory scores (F (1, 101) = 3.166, $p = 0.078 > 0.05$). Thus, this study excluded the influence of the exploratory pre-test score and tested whether there was a significant difference in the exploratory post-test scores of CDBL-TD and CDBL-BU students after Scratch training. Leneve's test indicated that the error variance of the two groups of dependent variables was not significantly different and was homogeneous (F = 0.041, $p = 0.839 > 0.05$). Results from a one-way ANCOVA indicate a significant difference between two groups' post-test scores (F (1, 101) = 12.684, $p = 0.001 < 0.05$, η2 = 0.112). The CDBL-TD adopted group mean (M = 9.817, SD = 0.304) is significantly higher than the CDBL-BU adopted group mean (M = 8.269, SD = 0.310). This result showed that, when using a CDBL-TD code teaching method to design a new Scratch project, the CDBL-TD exploratory score is better than that of the CDBL-BU group (Table 3).

Finally, we check whether statistical significance exists between groups and originality pre-test scores. The result shows that the within-group homogeneity of regression has no significance between groups and the originality pre-test score (F (1, 101) = 1.626, $p = 0.205 > 0.05$). Therefore, we excluded the influence of the originality pre-test score and tested whether there was a significant difference in the originality post-test scores of CDBL-TD and CDBL-BU students after Scratch training. Results from a one-way ANCOVA indicate that there was no significant difference between the two groups' post-test scores (F (1, 101) = 0.010, $p = 0.920 > 0.05$, η2 = 0.052) (Table 3).

3.2. Comparison of Three Design Dimensions and Scratch Project Design Scores

While the overarching dimensions of completeness, exploratory, and originality provide a macroscopic view of the students' proficiencies, dissecting each dimension further to obtain a microscopic understanding is essential. This in-depth examination of the sub-components of each primary dimension will shed light on the specific areas where each teaching methodology excels or falls short. Such a meticulous breakdown provides educators with detailed insights for potential pedagogical adjustments and highlights areas where students can focus their efforts. Therefore, in the subsequent sections, we dive deep into the individual competencies under each primary dimension, comparing the performance of both CDBL-TD and CDBL-BU groups. The details of the results are presented in Table 4. In the following table, we provide an analysis of the performance outcomes of the CDBL-TD and CDBL-BU groups.

Table 4. ANCOVA results of two teaching methods on three dimensions scores.

	Source of Variance	SS	Df	MS	F	η2
Completeness						
Completion degree	group	6.552	1	6.552	8.901 **	0.081
	pre-test	12.902	1	12.902	17.527 ***	0.148
	residual	74.350	101	0.736		
Simplicity	group	4.532	1	4.532	9.236 **	0.084
	pre-test	3.943	1	3.943	8.037 ***	0.074
	residual	49.555	101	0.491		
Correctness	group	9.809	1	9.809	18.692 ***	0.156
	pre-test	14.153	1	14.153	26.969 ***	0.211
	residual	53.003	101	0.525		
Exploratory						
Try different stages	group	24.376	1	24.376	12.493 **	0.110
	pre-test	5.912	1	5.912	3.030	0.029
	residual	197.069	101			
Creating different character types	group	0.032	1	0.032	0.041	0.000
	pre-test	5.574	1	5.574	7.809 *	0.066
	residual	79.418	101	0.786		
Using no-taught components	group	9.352	1	9.352	12.060 **	0.107
	pre-test	0.941	1	0.941	1.214	0.012
	residual	78.317	101	0.775		
Originality						
Game scenario complexity	group	12.795	1	12.795	27.304 ***	0.214
	pre-test	2.009	1	2.009	4.288 *	0.041
	residual	46.859	101	0.469		
Introduction of novel clearance conditions	group	0.938	1	0.938	1.021	0.010
	pre-test	15.016	1	15.016	16.346 ***	0.139
	residual	92.781	101	0.919		
Creation of distinctive roles	group	11.282	1	11.282	24.051 ***	0.194
	pre-test	1.962	1	1.962	4.183 *	0.040
	residual	46.907	101	0.469		

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

1. Completeness

First, in the "completion degree" dimension, the CDBL-TD group showed a significant advantage over the CDBL-BU group, as indicated by a one-way ANCOVA result ($F(1, 101) = 8.901$, $p = 0.004 < 0.05$, $\eta 2 = 0.081$). The CDBL-TD adopted group mean is significantly higher than the CDBL-BU adopted group mean ($M = 4.203$, $SD = 0.119$ versus $M = 3.691$, $SD = 0.121$). This result showed that, when using different coding teaching to design a new game project, the CDBL-TD completion degree score is better than that of the CDBL-BU group.

The "simplicity" dimension was also examined. Although both teaching methods seemed practical, the CDBL-TD method was found to be significantly more beneficial ($F(1, 101) = 9.236$, $p = 0.003 < 0.05$, $\eta 2 = 0.084$). The CDBL-TD adopted group means were significantly higher than the CDBL-BU adopted group mean ($M = 3.906$, $SD = 0.098$ versus $M = 3.470$, $SD = 0.100$). This result showed that using different coding teaching to design a new game project, the CDBL-TD simplicity score was better than that of the CDBL-BU group.

Finally, we investigated the "correctness" dimension. The CDBL-TD group also demonstrated superior results in terms of correctness, as supported by a significant ANCOVA outcome ($F(1, 101) = 18.692$, $p = 0.000 < 0.05$, $\eta 2 = 0.156$). The CDBL-TD adopted group mean is significantly higher than the CDBL-BU adopted group mean ($M = 3.993$, $SD = 0.101$ versus $M = 3.361$, $SD = 0.103$). This result showed that, by using different coding teaching methods to design a new game project, the CDBL-TD correctness score is better than that of the CDBL-BU group (Table 4).

2. Exploratory

We examined the "try different stages" dimension at first. No significant interaction was observed, but there was an indication that group variances might not be equal, suggesting differences in how students responded to the teaching methods.

Next, the dimension of "creating different character types" was investigated. Leneve's test indicated that the error variance of the two groups of dependent variables was not significantly different and was homogeneous ($F = 0.001$, $p = 0.981 > 0.05$). Results from a one-way ANCOVA indicate that there was no significant difference between two groups' post-test scores ($F(1, 101) = 0.041$, $p = 0.840 > 0.05$, $\eta 2 = 0.000$). No significant difference was detected between the groups, indicating that both methods were comparably effective in this aspect.

"Using no-taught components" was the last issue discussed. The within-group homogeneity of regression showed no significance between groups# and total pre-test scores ($F(1, 101) = 0.095$, $p = 0.759 > 0.05$). Therefore, this study excluded the influence of the pre-test score and tested whether there was a significant difference in the post-test scores of CDBL-TD and CDBL-BU students after Scratch training. Leneve's test indicated that the error variance of the two groups of dependent variables was not significantly different and was homogeneous ($F = 3.628$, $p = 0.06 > 0.05$). The results from a one-way ANCOVA indicate a significant difference between two groups' post-test scores ($F(1, 101) = 12.060$, $p = 0.001 < 0.05$, $\eta 2 = 0.107$). The CDBL-TD adopted group mean is significantly higher than the CDBL-BU adopted group mean ($M = 1.950$, $SD = 0.121$ versus $M = 1.346$, $SD = 0.124$). This result showed that, when using different Scratch teaching methods to design a new Scratch code project, the CDBL-TD score achieved is better than that of the CDBL-BU group (Table 4).

3. Originality

First, the "game scenario complexity" dimension was explored. The results suggested a significant difference in group variances, indicating varied student responses to the teaching strategies. Next, we discussed the "introduction of novel clearance conditions" dimension. The two methods were comparable in this dimension, with no significant difference observed. Finally, the "creation of distinctive roles" was investigated. The within-group homogeneity of regression showed no significance between groups and total

pre-test scores (F (1, 101) = 0.093, p = 0.761 > 0.05). Therefore, this study excluded the influence of the pre-test score and tested whether there was a significant difference in the post-test scores of CDBL-TD and CDBL-BU students after Scratch training. Leneve's test indicated that the error variance of the two groups of dependent variables was not significantly different and was homogeneous (F = 0.008, p = 0.929 > 0.05). The results from a one-way ANCOVA indicate a significant difference between the two groups' post-test scores (F (1, 101) = 24.51, p = 0.000 < 0.05, η2 = 0.194). The CDBL-TD adopted group mean is significantly higher than the CDBL-BU adopted group mean (M = 2.911, SD = 0.095 versus M = 2.247, SD = 0.096). The CDBL-TD method was significantly more effective than the CDBL-BU method (Table 4).

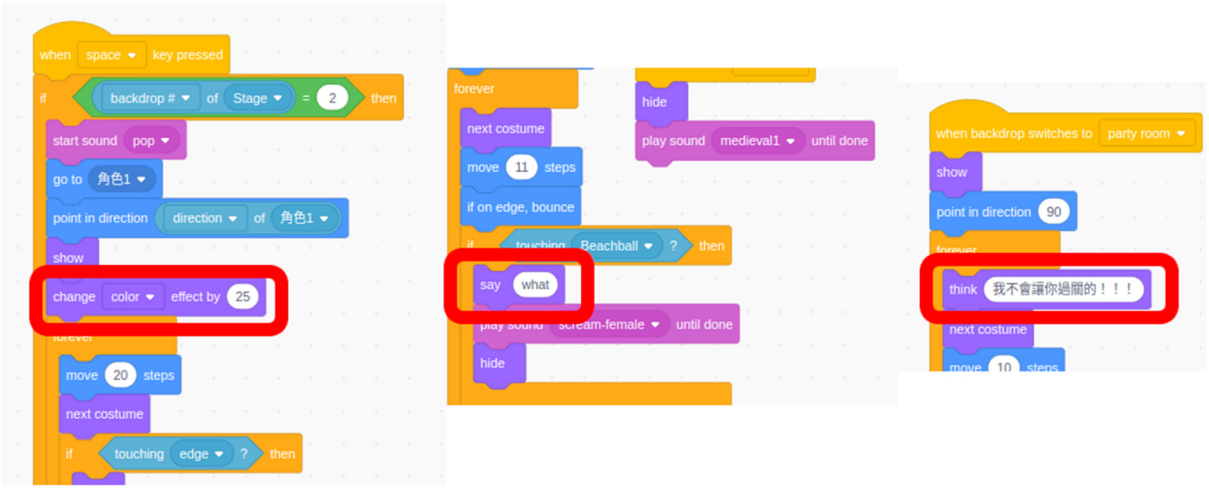
4. Discussion

The primary aim of this study was to examine the impact of two distinct learning scaffoldings, CDBL-TD and CDBL-BU, in terms of improving students' Scratch programming skills, exploratory learning, and creative application. Our thorough evaluation provided insights into how both learning scaffoldings influenced students' self-regulated learning.

The first research question determined the effects of different learning scaffoldings on fifth-grade programming beginners' programming ability and creativity. The CDBL-TD group exhibited a significant advantage in overall post-test scores. This discrepancy suggests that the CDBL-TD learning scaffold affords students with better opportunities for self-regulated exploratory learning, encompassing a broader range of skills, particularly in the foundational areas of Scratch programming. The CDBL-TD learning scaffolding aligns with the benefits of general domain scaffolding, wherein students grasp the main concepts through complete codes before delving into details [68], resulting in increased learning benefits.

When examining the 'completeness' dimension, it was found that providing the CDBL-TD group with a complete programming learning scaffolding, enabling students to autonomously deconstruct the program, significantly enhanced their programming abilities. The provision of complete codes allowed students to perceive the interconnections and details within the code, fostering a more comprehensive understanding rather than a fragmented one. Furthermore, in autonomous deconstruction, students could adjust their learning pace based on their abilities, avoiding hindrance by the traditional programming teaching process. This learning process contributed to improved programming skills and enhanced game quality. These findings were validated across the three sub-categories of completeness (game design completion, code conciseness, and game execution correctness). These results address research question RQ1 and reinforce the advantages of general domain scaffolding [68].

The second research question delves into how different learning scaffoldings can inspire students to autonomously utilize untaught Scratch functions. Discrepancies emerged when scrutinizing the "exploratory" dimension. In terms of overall exploration scores, the CDBL-TD group outperformed others. It was observed that students in this group exhibited a more comprehensive understanding of game design and displayed greater interest in exploring unknown aspects, a trend also substantiated in the "Using un-taught components" subcategory. Figure 5 shows that students in the CDBL-TD group each added three program functions, not taught by the teacher, in the advanced version of the shooting game. They tried things like changing the color of objects after being affected 25 times (Figure 5a), making objects speak prompts (Figure 5b), and setting prompts where the devil would prevent challengers from passing levels (Figure 5c). This discovery holds significant implications for fostering students' creativity.



(**a**) Change the effect (**b**) Say (**c**) Think

Figure 5. New functions found by students in the CDBL-TD group.

In creativity-related research, the emphasis has been on students' creative abilities [48,49]. However, generating original ideas necessitates supporting students' abilities, a feat that it is challenging to achieve solely through imagination, requiring the accumulation of learning experiences. Therefore, nurturing students' capacity to explore the unknown and actively broaden their learning experiences is a focal point in creativity-related research [40]. Compared to the CDBL-BU learning scaffolding, the CDBL-TD learning scaffolding provides a learning environment where learners can more actively explore unknown realms and freely experiment with various possibilities. These findings contribute to learning scaffolding for cultivating creativity and address research question RQ2.

The final research question explored potential differences in program innovation functionality and originality performance between the CDBL-TD and CDBL-BU learning scaffoldings. This study scrutinized "originality" and identified that, although there was no significant difference in the post-test scores between the two groups, the students' works demonstrated the equal effectiveness of both learning scaffoldings. The students' creations markedly deviated from the advanced shooter examples provided by the instructor. Both groups showcased student originality under different learning scaffolding designs. We compared the monotonous game pass screen of the example game provided by the teacher, the game end screen of CDBL-TD, and the game end screen of CDBL-BU (Figure 6). The game graphics of the latter two were more creative. However, in the subcategory of originality, specifically in the "creation of distinctive roles", the object characters designed by the CDBL-TD team were notably more refreshing. The game graphics of the CDBL-TD group incorporated more diverse characters and added pyrotechnic effects, and the CDBL-BU group only arranged the characters provided by Scratch. Please see Figure 6b,c.

(**a**) Teacher (**b**) CDBL-TD (**c**) CDBL-BU

Figure 6. The results of advanced shooting games from the different groups.

The study posited that, after students autonomously deconstruct the complete code, they acquired a comprehensive and broad understanding of game design examples, facilitating the creation of unique characters to enhance their own games. The CDBL-TD learning scaffolding is deemed more likely to preserve learners' imaginative space than the CDBL-BU learning scaffolding. This discovery also addresses our research question RQ3 and prompts further in-depth research to explore this issue.

A crucial element of self-regulated learning scaffolding hinges on individualized adaptability, timeliness, and calibrated feedback [43,44]. Personalized learning scaffolding proves essential as self-regulated learning unfolds to the degree that individual learners employ metacognition and learning strategies [47]. Self-regulation processes are acquired and scaffolded through guidance and modeling from others [46]. The two learning scaffoldings proposed in this study, CDBL-TD and CDBL-BU, allow programming beginners to disassemble the program code to learn independently. Because students' learning is active, students can focus more on improving their programming abilities through applying CT skills. Therefore, providing programming teaching for beginners provides a different way. In addition, these two learning scaffoldings allow students to adjust their learning progress according to their own needs. This part provides different solutions for the gap in traditional program teaching progress. Under these two learning scaffolds, students can self-regulate and effectively arrange their study time, improving their academic performance, which aligns with the self-regulated learning theory [45]. Under the condition of having a sense of learning achievement, students improve their learning motivation and reduce the cognitive load of the learning program, thereby also improving the expression of students' creativity. Furthermore, the game creation process requires continuity without interruption, and creativity can be continuously triggered. This study also found these creative effects in the students' creative games.

The findings of this study indicate that furnishing complete learning scaffoldings for students' self-regulated exploratory learning is valuable, contributing to enhanced learning effectiveness and the fostering and manifestation of creativity. Traditional teaching paces disregarding individual differences often fail to deliver personalized learning experiences, leading advanced students to plateau and less capable students to persistently lag. The authors of this study advocate for further research dedicated to exploring and developing self-learning scaffoldings.

5. Limitations

Despite this study's aim to provide an empirical basis for the theoretical connection between complete self-regulated learning exploration scaffoldings and student game design performance, certain limitations must be considered. This study's limitation is the lack of a control group with which to reduce the impact of uncertain variables in the experiment, which makes it challenging to study the effects of the two scaffolding learning methods. In the future, it is recommended that a control group be added to the Scratch programming teaching to enhance the experiment's credibility. The specific age group focused on 11- to 12-year-old Scratch beginners might not represent the potential responses of different age groups or proficiency levels. The cultural and regional background may also limit the generalizability of our conclusions. Furthermore, the limited sample scope could weaken the robustness of our results. The significant improvements in learning outcomes and motivation indicate that the CDBL-TD and CDBL-BU program-learning scaffolding methods are appropriate for elementary school students; however, we expect to better support computational thinking skills in decomposition.

6. Conclusions and Future Work

The CDBL-TD self-regulated learning scaffolding proposed in our study fully embodies the advantages of domain-general scaffolding [68], allowing students to obtain more detailed learning benefits by autonomously deconstructing complete codes. Moreover, our design adopts a reverse-engineering approach for deconstructing learning, enhancing

students' understanding of the content [39,40]. A key finding is that students demonstrated higher motivation to explore the unknown, aligning with findings from other reverse-engineering studies [36,37].

Regarding creative expression, both CDBL-TD and CDBL-BU learning scaffoldings showed comparable progress. Through the autonomous process of deconstructing codes, students could explore learning at their own pace without being forced to alter their learning steps to match the class's pace. This uninterrupted flow of creativity allowed for more original outputs [59–61], akin to poets producing beautiful verses when their thoughts flow unhindered. These findings underscore the importance of hands-on STEM activities [1].

Despite the extensive research on learning scaffoldings across various fields, studies supporting self-regulated learning scaffoldings for programming novices are still limited. The question of equip frontline teachers with the tools to nurture students' programming and creative application skills requires further research. This study provides a foundation and new thinking direction for such research. However, it is too early to conclude that the learning method of CDBL-TD is better than CDBL-BU. Many research aspects can be considered in addition, such as motivation, prior experience with similar tools, different instructions, and the assistance of AI tools. From a long-term perspective, besides incorporating a more diverse sample size, other age groups, emphasizing the long-term effects of these teaching methods on student outcomes and creativity, might further enrich academic discourse in this field. Additionally, our research outcomes can serve as a reference for developing self-regulated learning in other areas, paving the way for more in-depth studies.

Author Contributions: Conceptualization, S.-W.S. and C.-T.S.; methodology, S.-W.S. and L.-X.C.; validation, S.-W.S. and L.-X.C.; formal analysis, S.-W.S.; investigation, L.-X.C.; resources, L.-X.C.; data curation, S.-W.S. and L.-X.C.; writing—original draft preparation, S.-W.S.; writing—review and editing, S.-W.S. and L.-X.C.; visualization, S.-W.S.; supervision, S.-M.Y. and C.-T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data will be available from corresponding authors on a reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Darmawansah, D.; Hwang, G.-J.; Chen, M.-R.A.; Liang, J.-C. Trends and research foci of robotics-based STEM education: A systematic review from diverse angles based on the technology-based learning model. *Int. J. STEM Educ.* **2023**, *10*, 12. [CrossRef]
2. Lee, A. Determining the effects of computer science education at the secondary level on STEM major choices in postsecondary institutions in the United States. *Comput. Educ.* **2015**, *88*, 241–255. [CrossRef]
3. Ryoo, J.J.; Margolis, J.; Lee, C.H.; Sandoval, C.D.; Goode, J. Democratizing computer science knowledge: Transforming the face of computer science through public high school education. *Learn. Media Technol.* **2013**, *38*, 161–181. [CrossRef]
4. Chou, C.-H.; Su, Y.-S.; Chen, H.-J. Interactive teaching aids integrating building blocks and programming logic. *J. Internet Technol.* **2019**, *20*, 1709–1720.
5. Resnick, M.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner, A.; Rosenbaum, E.; Silver, J.; Silverman, B. Scratch: Programming for all. *Commun. ACM* **2009**, *52*, 60–67. [CrossRef]
6. Su, A.Y.; Huang, C.S.; Yang, S.J.; Ding, T.-J.; Hsieh, Y. Effects of annotations and homework on learning achievement: An empirical study of Scratch programming pedagogy. *J. Educ. Technol. Soc.* **2015**, *18*, 331–343.
7. Wing, J.M. Computational thinking and thinking about computing. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2008**, *366*, 3717–3725. [CrossRef]
8. Wu, S.-Y.; Su, Y.-S. Visual programming environments and computational thinking performance of fifth-and sixth-grade students. *J. Educ. Comput. Res.* **2021**, *59*, 1075–1092. [CrossRef]
9. Noone, M.; Mooney, A. Visual and textual programming languages: A systematic review of the literature. *J. Comput. Educ.* **2018**, *5*, 149–174. [CrossRef]

10. Tsai, C.-Y. Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Comput. Hum. Behav.* **2019**, *95*, 224–232. [CrossRef]
11. Tsukamoto, H.; Takemura, Y.; Oomori, Y.; Ikeda, I.; Nagumo, H.; Monden, A.; Matsumoto, K.-I. Textual vs. visual programming languages in programming education for primary schoolchildren. In Proceedings of the 2016 IEEE Frontiers in Education Conference (FIE), Erie, PA, USA, 12–15 October 2016; pp. 1–7.
12. Kelleher, C.; Pausch, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv. (CSUR)* **2005**, *37*, 83–137. [CrossRef]
13. Weintrop, D. Blocks, text, and the space between: The role of representations in novice programming environments. In Proceedings of the 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Atlanta, GA, USA, 18–22 October 2015; pp. 301–302.
14. Feijóo-García, P.G.; Kapoor, A.; Gardner-McCune, C.; Ragan, E. Effects of a block-based scaffolded tool on students' introduction to hierarchical data structures. *IEEE Trans. Educ.* **2021**, *65*, 191–199. [CrossRef]
15. Jung, I.; Choi, J.; Kim, I.-J.; Choi, C. Interactive learning environment for practical programming language based on web service. In Proceedings of the 2016 15th International Conference on Information Technology Based Higher Education and Training (ITHET), Istanbul, Turkey, 8–10 September 2016; pp. 1–7.
16. Kalelioğlu, F. A new way of teaching programming skills to K-12 students: Code. org. *Comput. Hum. Behav.* **2015**, *52*, 200–210. [CrossRef]
17. Park, K.; Mott, B.; Lee, S.; Glazewski, K.; Scribner, J.A.; Ottenbreit-Leftwich, A.; Hmelo-Silver, C.E.; Lester, J. Designing a visual interface for elementary students to formulate ai planning tasks. In Proceedings of the 2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), St Louis, MO, USA, 10–13 October 2021; pp. 1–9.
18. Brennan, K.; Resnick, M. New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, BC, Canada, 13–17 April 2012; p. 25.
19. Brennan, K.; Resnick, M. Stories from the scratch community: Connecting with ideas, interests, and people. In Proceedings of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO, USA, 6–9 March 2013; pp. 463–464.
20. Dann, W.; Cosgrove, D.; Slater, D.; Culyba, D.; Cooper, S. Mediated transfer: Alice 3 to java. In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, Raleigh, NC, USA, 29 February–3 March 2012; pp. 141–146.
21. Moskal, B.; Lurie, D.; Cooper, S. Evaluating the effectiveness of a new instructional approach. In Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, VA, USA, 3–7 March 2004; pp. 75–79.
22. Du, J.; Wimmer, H.; Rada, R. "Hour of Code": Can It Change Students' Attitudes toward Programming? *J. Inf. Technol. Educ. Innov. Pract.* **2016**, *15*, 53. [CrossRef]
23. Patton, E.W.; Tissenbaum, M.; Harunani, F. MIT app inventor: Objectives, design, and development. In *Computational Thinking Education*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 31–49.
24. Pokress, S.C.; Veiga, J.J.D. MIT App Inventor: Enabling personal mobile computing. *arXiv* **2013**, arXiv:1310.2830.
25. Thuneberg, H.; Salmi, H.; Bogner, F.X. How creativity, autonomy and visual reasoning contribute to cognitive learning in a STEAM hands-on inquiry-based math module. *Think. Ski. Creat.* **2018**, *29*, 153–160. [CrossRef]
26. Giomataris, I. Georges Charpak (1924–2010). *Nature* **2010**, *467*, 1048. [CrossRef]
27. Su, S.-W.; Jung, S.-Y.; Yu, X.; Yuan, S.-M.; Sun, C.-T. Modify, Decompose and Reassemble: Learner-Centered Constructive Teaching Strategy for Introductory Programming Course in College. In Proceedings of the 2022 IEEE 5th Eurasian Conference on Educational Innovation (ECEI), Taipei, Taiwan, 10–12 February 2022; pp. 197–200.
28. McGowan, V.C.; Ventura, M.; Bell, P. Reverse Engineering. *Sci. Child.* **2017**, *54*, 68. [CrossRef]
29. Chikofsky, E.J.; Cross, J.H. Reverse engineering and design recovery: A taxonomy. *IEEE Softw.* **1990**, *7*, 13–17. [CrossRef]
30. West, A.B.; Sickel, A.J.; Cribbs, J.D. The science of solubility: Using reverse engineering to brew a perfect cup of coffee. *Sci. Act.* **2015**, *52*, 65–73. [CrossRef]
31. Lee, K.H.; Woo, H. Use of reverse engineering method for rapid product development. *Comput. Ind. Eng.* **1998**, *35*, 21–24. [CrossRef]
32. Elizalde, H.; Rivera-Solorio, I.; Perez, Y.; Morales-Menéndez, R.; Orta, P.; Guerra, D.; Ramirez, R.A. An educational framework based on collaborative reverse engineering and active learning: A case study. *Int. J. Eng. Educ.* **2008**, *24*, 1062.
33. Wood, K.L.; Jensen, D.; Bezdek, J.; Otto, K.N. Reverse engineering and redesign: Courses to incrementally and systematically teach design. *J. Eng. Educ.* **2001**, *90*, 363–374. [CrossRef]
34. Dempere, L.A. Reverse engineering as an educational tool for sustainability. In Proceedings of the 2009 IEEE International Symposium on Sustainable Systems and Technology, Tempe, AZ, USA, 18–20 May 2009; pp. 1–3.
35. Verner, I.; Greenholts, M. Teacher education to analyze and design systems through reverse engineering. In *Educational Robotics in the Makers Era*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 122–132.
36. Griffin, J.; Kaplan, E.; Burke, Q. Debug'ems and other deconstruction kits for STEM learning. In Proceedings of the IEEE 2nd Integrated STEM Education Conference, Ewing, NJ, USA, 9 March 2012; pp. 1–4.
37. Rogers-Chapman, M.F. Accessing STEM-focused education: Factors that contribute to the opportunity to attend STEM high schools across the United States. *Educ. Urban Soc.* **2014**, *46*, 716–737. [CrossRef]

38. Klimek, I.; Keltika, M.; Jakab, F. Reverse engineering as an education tool in computer science. In Proceedings of the 2011 9th International Conference on Emerging eLearning Technologies and Applications (ICETA), Stara Lesna, Slovakia, 27–28 October 2011; pp. 123–126.
39. Chattopadhyay, S. Material and processing basics through reverse engineering. In Proceedings of the 2017 ASEE Annual Conference & Exposition, Columbus, OH, USA, 25–28 June 2017.
40. Shcherbakov, V.S.; Makarov, A.L.; Buldakova, N.V.; Butenko, T.y.P.; Fedorova, L.V.; Galoyan, A.R. Development of higher education students' creative abilities in learning and research activity. *Eurasian J. Anal. Chem.* **2017**, *12*, 765–778. [CrossRef]
41. Van Gog, T.; Hoogerheide, V.; Van Harsel, M. The role of mental effort in fostering self-regulated learning with problem-solving tasks. *Educ. Psychol. Rev.* **2020**, *32*, 1055–1072. [CrossRef]
42. Zimmerman, B.J. Attaining self-regulation: A social cognitive perspective. In *Handbook of Self-Regulation*; Elsevier: Amsterdam, The Netherlands, 2000; pp. 13–39.
43. Azevedo, R.; Hadwin, A.F. Scaffolding self-regulated learning and metacognition–Implications for the design of computer-based scaffolds. *Instr. Sci.* **2005**, *33*, 367–379. [CrossRef]
44. Azevedo, R.; Moos, D.C.; Greene, J.A.; Winters, F.I.; Cromley, J.G. Why is externally-facilitated regulated learning more effective than self-regulated learning with hypermedia? *Educ. Technol. Res. Dev.* **2008**, *56*, 45–72. [CrossRef]
45. Wolters, C.A.; Won, S.; Hussain, M. Examining the relations of time management and procrastination within a model of self-regulated learning. *Metacognition Learn.* **2017**, *12*, 381–399. [CrossRef]
46. Zimmerman, B.J. Becoming a self-regulated learner: An overview. *Theory Into Pract.* **2002**, *41*, 64–70. [CrossRef]
47. Zimmerman, B.J. A social cognitive view of self-regulated academic learning. *J. Educ. Psychol.* **1989**, *81*, 329. [CrossRef]
48. Gangadharbatla, H. Technology component: A modified systems approach to creative thought. *Creat. Res. J.* **2010**, *22*, 219–227. [CrossRef]
49. Daud, S.K.M.; Mustaffa, F.; Hussain, H.; Osman, M.N. Creative technology as open ended learning tool: A case study of design school in Malaysia. *Int. J. Educ. Pedagog. Sci.* **2009**, *3*, 1839–1842.
50. Chen, C.-H.; Chiu, C.-H. Employing intergroup competition in multitouch design-based learning to foster student engagement, learning achievement, and creativity. *Comput. Educ.* **2016**, *103*, 99–113. [CrossRef]
51. Piaget, J. *Construction of Reality in the Child*; Routledge & Kegan Paul: London, UK, 1957; pp. 80–86.
52. Kao, G.Y.-M.; Chiang, C.-H.; Sun, C.-T. Customizing scaffolds for game-based learning in physics: Impacts on knowledge acquisition and game design creativity. *Comput. Educ.* **2017**, *113*, 294–312. [CrossRef]
53. Stoyle, K.L.; Morris, B.J. Blogging mathematics: Using technology to support mathematical explanations for learning fractions. *Comput. Educ.* **2017**, *111*, 114–127. [CrossRef]
54. Gee, J.P. What video games have to teach us about learning and literacy. *Comput. Entertain. (CIE)* **2003**, *1*, 20. [CrossRef]
55. Navarrete, C.C. Creative thinking in digital game design and development: A case study. *Comput. Educ.* **2013**, *69*, 320–331. [CrossRef]
56. Caperton, I.H. Toward a theory of game-media literacy: Playing and building as reading and writing. *Int. J. Gaming Comput.-Mediat. Simul. (IJGCMS)* **2010**, *2*, 1–16. [CrossRef]
57. Ke, F. An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Comput. Educ.* **2014**, *73*, 26–39. [CrossRef]
58. Sung, H.-Y.; Hwang, G.-J.; Lin, C.-J.; Hong, T.-W. Experiencing the Analects of Confucius: An experiential game-based learning approach to promoting students' motivation and conception of learning. *Comput. Educ.* **2017**, *110*, 143–153. [CrossRef]
59. Flumerfelt, S.; Green, G. Using lean in the flipped classroom for at risk students. *J. Educ. Technol. Soc.* **2013**, *16*, 356–366.
60. Fulton, K.P. 10 reasons to flip. *Phi Delta Kappan* **2012**, *94*, 20–24. [CrossRef]
61. Vandewaetere, M.; Clarebout, G. Can instruction as such affect learning? The case of learner control. *Comput. Educ.* **2011**, *57*, 2322–2332. [CrossRef]
62. Kurkovsky, S. Teaching software engineering with LEGO Serious Play. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, Vilnius, Lithuania, 4–8 July 2015; pp. 213–218.
63. Bidanda, B.; Motavalli, S.; Harding, K. Reverse engineering: An evaluation of prospective non-contact technologies and applications in manufacturing systems. *Int. J. Comput. Integr. Manuf.* **1991**, *4*, 145–156. [CrossRef]
64. Rosner, D.K.; Ames, M. Designing for repair? Infrastructures and materialities of breakdown. In Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, Baltimore, MD, USA, 15–19 February 2014; pp. 319–331.
65. Abdüsselam, M.S.; Turan-Güntepe, E.; Durukan, Ü.G. Programming education in the frameworks of reverse engineering and theory of didactical situations. *Educ. Inf. Technol.* **2022**, *27*, 6513–6532. [CrossRef]
66. Zhang, J.-H.; Meng, B.; Zou, L.-C.; Zhu, Y.; Hwang, G.-J. Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy. *Interact. Learn. Environ.* **2023**, *31*, 3792–3809. [CrossRef]
67. Edwards, J.; Ditton, J.; Trninic, D.; Swanson, H.; Sullivan, S.; Mano, C. Syntax exercises in CS1. In Proceedings of the 2020 ACM Conference on International Computing Education Research, Dunedin, New Zealand, 8–13 August 2020; pp. 216–226.
68. McNeill, K.L.; Krajcik, J. Supporting students' construction of scientific explanation through generic versus context-specific written scaffolds. In Proceedings of the Annual Meeting of the American Educational Research Association, San Francisco, CA, USA, 7–11 April 2006.

69. Lee, H.-S.; Songer, N.B. Expanding an understanding of scaffolding theory using an inquiry-fostering science program. *J. Learn. Sci.* **2004**, *15*, 153–191.
70. Perkins, D.N.; Salomon, G. Are cognitive skills context-bound? *Educ. Res.* **1989**, *18*, 16–25. [CrossRef]
71. Bibi, A.; Zamri, S.N.A.S.; Abedalaziz, N.A.M.; Ahmad, M.; Dad, H. Role of Students' Context Familiarity in Differential Equations Problem Solving at Pre University Level. *MOJES Malays. Online J. Educ. Sci.* **2018**, *6*, 48–57.
72. Amabile, T.M. The social psychology of creativity: A componential conceptualization. *J. Personal. Soc. Psychol.* **1983**, *45*, 357. [CrossRef]
73. Armstrong, R.L. The midpoint on a five-point Likert-type scale. *Percept. Mot. Ski.* **1987**, *64*, 359–362. [CrossRef]
74. Rijo-García, S.; Segredo, E.; León, C. Computational thinking and user interfaces: A systematic review. *IEEE Trans. Educ.* **2022**, *65*, 647–656. [CrossRef]
75. Guilford, J.P. Intelligence: 1965 model. *Am. Psychol.* **1966**, *21*, 20. [CrossRef]