computers

Article

Processing Analysis of Swift Playgrounds in a Children's Computational Thinking Course to Learn Programming

Guo-Ming Cheng * and Chia-Pin Chen

Department of Industry Technology Education, National Kaohsiung Normal University, 62, Shenjhong Rd., Yanchao District, Kaohsiung 82446, Taiwan; chen.c0902@gmail.com
* Correspondence: t3791@mail.nknu.edu.tw

Abstract: Computational thinking courses can cultivate students' ability to apply logic in the fields of mathematics and information science. The new 12-year Basic Education Curriculum Guidelines were implemented in Fall 2019 in Taiwan. Courses on computational thinking, problem solving, and programming are contained in the technology education field in junior and senior high schools. Swift Playgrounds is an innovative app for the iPad and Mac that makes learning Swift interactive and fun. No programming knowledge is required to use Swift Playgrounds, making it very suitable for beginners. This study was carried out by letting elementary school teachers and students participate in Swift Playgrounds computational thinking courses. By trying this app, teachers of different disciplines attempted to realize more learning situations. Students learned how to cope with functions and loop skills by playing with "Byte", which is a character in Swift Playgrounds. There were three purposes for this study: first, designing a computational thinking course for the most basic part, "Hello! Byte", in Swift Playgrounds; second, assigning elementary school teachers to assess the qualitative analysis of tasks in Swift Playgrounds; and third, assigning elementary school students to do the tasks and assign a difficulty index in Swift Playgrounds after learning with this app. The results show that most teachers considered this approach to be able to improve logical thinking and inferential capability after assessing, and most students considered functions and loops quite difficult after using the app. According to the students' indices, about 86 percent of students considered that adding commands is easy, and about 37 percent of students considered that functions are easy. On the other hand, about 24 percent of students considered that applying the Slotted Stairways is difficult, and about 34 percent of students considered that using loops is hard. It is suggested that more instructions for the course or extendibility for classes is required.

Keywords: computational thinking; Swift Playgrounds; 12-year Basic Education; Bebras; programming

check for updates

Citation: Cheng, G.-M.; Chen, C.-P. Processing Analysis of Swift Playgrounds in a Children's Computational Thinking Course to Learn Programming. *Computers* **2021**, *10*, 68. https://doi.org/10.3390/computers10050068

Academic Editors: Carlos Vaz de Carvalho and Antonio Coelho

Received: 30 March 2021
Accepted: 10 May 2021
Published: 20 May 2021

1. Introduction

Computational thinking through programming is attracting increased attention, as it is considered an ideal pathway for the development of 21st-century skills; this has led to K-12 initiatives around the world and a rapid increase in relevant research studies [1,2]. Computational thinking is considered an ideal skill for future development [3,4]. Educating future generations in programming and computational thinking is not trivial, and many different platforms and teaching approaches can be used for this purpose [5–7]. Swift is one tool for learning programming, and it is a development tool specially designed for designing iOS applications [8,9]. Swift Playgrounds, announced at the Apple Worldwide Developers Conference (WWDC) in June 2016, is an innovative and powerful app and an exceptionally simple way to build user interfaces across all Apple platforms using the power of Swift. It provides several-hour programming courses, suitable for children and beginners learning programming, and can build user interfaces for any Apple device using just one set of tools and APIs. Beginners can grasp the basic concept of using Swift through tasks, and the strong multitouch function allows easier learning of programming with Swift

Playgrounds. Simply by touching and dragging commands or inputting text and numbers, the users can interact with the game's role for programming and further learn the basic and solid grammar components of Swift, such as functions, loops, variables, parameters, and arrays [10].

Computational thinking is becoming more important in global information science and information curricula, and methods for including it in curricula are being sought [11–13]. More than 50 countries now participate in the Bebras challenge, which began in 2004. Its thematic short questions allow students from elementary schools through to senior high schools to solve problems online; the problem-solving time for each is about 3–5 min. Some computational thinking skills, e.g., mathematics, abstract making, computational thinking, problem solving, and estimation and induction, are also included. Bebras questions cover algorithms, data structures, programming, the Internet, databases, and social and moral issues [14].

In the experimental class in this study, 29 G5 students attended the 2018 Bebras International Challenge on Informatics and Computational Thinking in the first term and participated in the Swift Playgrounds computational thinking curriculum in the second term of the 2018 academic year. Practice with Bebras questions could train students' computational thinking capabilities, including programming capability, problem solving skills, decomposition of complicated tasks into simple components, algorithm design, and pattern recognition, to conform to the Curriculum Guidelines of 12-Year Basic Education—Technology, covering data representation, processing, analysis, algorithms, and information technology applications [15].

Consequently, this study aimed to (1) design a six-session Swift Playgrounds iPad app computational thinking course for elementary schools, (2) arrange for nine elementary school teachers to assess the tasks in the Swift Playgrounds iPad app and to provide qualitative analysis, and (3) arrange for 29 elementary school G5 students to provide difficulty analyses of task learning with the Swift Playgrounds iPad app.

2. Literature Review

Computational thinking [16,17] includes data collection, data analysis, pattern searching, abstract making, data resolution, modeling, and algorithms. Computational thinking can be applied in real life to break down problems, make complicated problems into simpler ones, and follow the context to solve problems and gain more information [18,19]. Its application to each subject is similar to including computational thinking in the technology field, in 12-year Basic Education [15]. A transnational study on robotics education between China and the USA developed a tool to evaluate elementary school G5 students' computational thinking capability, to assist students in learning problem challenges and computational thinking capability [20]. The Swedish government introduced digital computational thinking capability training courses and included them in the K-9 programming curriculum in 2018. More than 100,000 teachers had to learn programming and computational thinking instruction in a short period [21]. Such a changing trend of thought is unprecedented; even the 2019 12-year Basic Education Curriculum Guidelines in Taiwan stressed the teaching of a computational thinking curriculum.

For the challenge of computational thinking, the Italy Bebras official website [22] has provided services to teachers and students since 2015 to support task preparation and train students in solving problems; it manages about 25,000 teams and training courses. Lithuania and the UK have supported curriculum teaching and practice for the Bebras challenge, using the Bebras platform [14] to encourage students in information technology and computational thinking and educators in taking the computational thinking syllabus into account. The Bebras challenge provides creative and interesting tasks. Previous research [23] analyzed the Bebras task performance of 115,400 G3–G12 students in Italy, Australia, Finland, Lithuania, South Africa, Switzerland, and Canada; Bebras task performance data were collected and analyzed to reflect learning in computational thinking challenges. Algorithm and data representation questions dominated the performance of challenge tasks, comprising about 75–90%. For this reason, when providing teachers with a computational thinking curriculum, algorithm and data representation questions could be listed as the main points, and abstract, parallel, and question resolution items should be supplemental [24]. The author of [25] arranged for elementary school G5 students to participate in the 2017 Bebras International computational thinking challenge and discussed questions for elementary school students via Padlet and team discussion; the technology acceptance model tool was used for 333 students filling in feedback on the "perceived usefulness" of Padlet, and 74.4% of them considered it helpful.

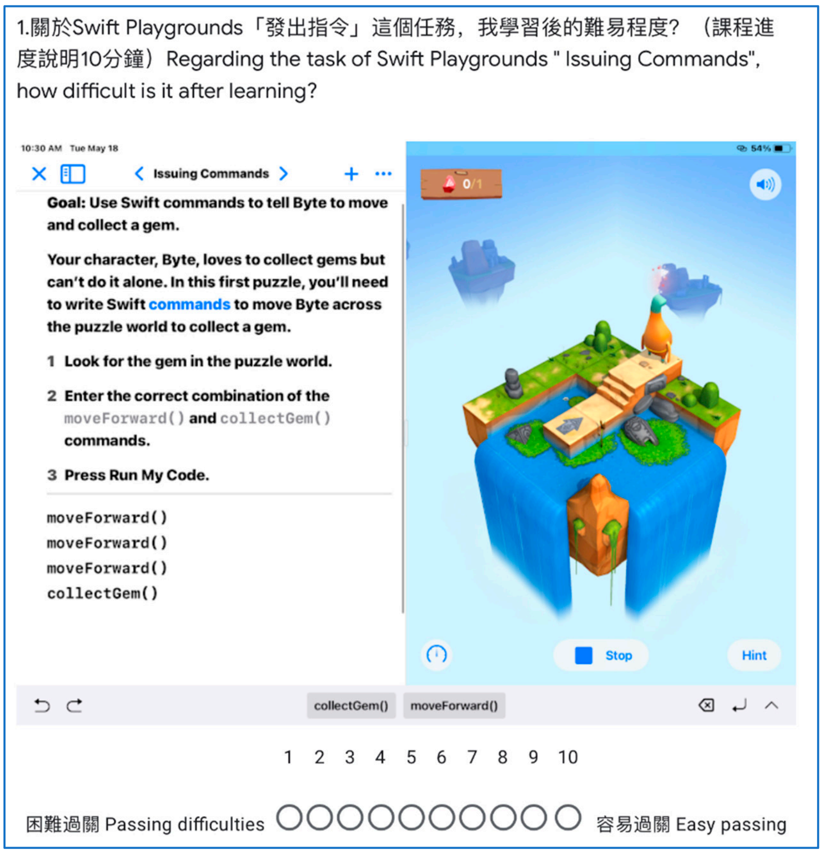
In the Everyone Can Code plan in Chicago [26], the curriculum in the full-featured app was designed by Apple, allowing students to construct personal designs by exploring basic coding concepts. It provided all G3–G12 students with opportunities for coding education, as well as volunteers and students with opportunities for practicing programming in local enterprises to expand opportunities for students cultivating coding skills and inquiring into career development. KIBO's programming kit [27] was composed of 21 unique cards to assemble complicated sequences, including loops and conditional and embedded statements. Furthermore, in order to enhance interdisciplinary integration of STEAM, the tool contained various art creation materials for children making personalized products. Falloon indicated in research in 2016 [28] that the Scratch Jnr coding curriculum for students aged 5 and 6 in New Zealand provided an important method to train students in complicated computational thinking and critical thinking ability, and it provided critical evidence for teachers of the students' thinking processes in computational tasks. Regarding the coding curriculum in elementary schools in Italy [29], vocational high school students, in the theoretical framework provided in computational thinking, taught junior high school and elementary school students to use the App Inventor to create apps on smartphones in an Android environment; this formed an interesting cooperation pattern between elementary schools and high schools.

3. Research Method and Results

A survey research method was utilized in this study. The researcher instructed a G5 computer class. The designed teaching process contained 6 sessions, with 1 session (40 min) per week practiced in the computer class. Nine teachers from different fields were invited to try out and assess the "Hello! Byte" computational thinking curriculum on Swift Playgrounds, and 29 students learnt the "Hello! Byte" computational thinking course on Swift Playgrounds. After participating in the experiential learning, teachers and students responded to a Google form to explain their qualitative analysis and difficulty analysis of the computational thinking curriculum. In Figure 1, the Google Form of the feedback for the degree of difficulty is shown in the screenshot. In Figure 2, the screenshot on the left is the role of "Byte" in the Swift Playground app, and the one on the right presents the scene of the task for the coding game.

3.1. Teaching Process Design and Feedback Analysis after Students' Learning of Swift Playgrounds Computational Thinking

A Google form was used to collect difficulty feedback from the 29 elementary school G5 students after they learned the Swift Playgrounds computational thinking curriculum, from Task I to Task VIII, for six sessions (240 min). The feedback was analyzed using a Google form linear scale (the most difficult tasks were given a score of 1, the easiest tasks were given 10). The researcher proposed a linear difficulty scale of 1–3 as difficult, 4–7 as moderate, and 8–10 as easy, as shown in Figures 3–12.



1.關於Swift Playgrounds「發出指令」這個任務，我學習後的難易程度? （課程進度說明10分鐘）Regarding the task of Swift Playgrounds " Issuing Commands", how difficult is it after learning?

10:30 AM Tue May 18

Issuing Commands

Goal: Use Swift commands to tell Byte to move and collect a gem.

Your character, Byte, loves to collect gems but can't do it alone. In this first puzzle, you'll need to write Swift commands to move Byte across the puzzle world to collect a gem.

1 **Look for the gem in the puzzle world.**

2 **Enter the correct combination of the** `moveForward()` **and** `collectGem()` **commands.**

3 **Press Run My Code.**

```
moveForward()
moveForward()
moveForward()
collectGem()
```

	1	2	3	4	5	6	7	8	9	10	
困難過關 Passing difficulties	○	○	○	○	○	○	○	○	○	○	容易過關 Easy passing

Figure 1. The Google Form of the feedback for the degree of difficulty was shown in the screenshot. The Chinese meaning of this picture is the feedback of the degree of difficulty for Task I "Issuing Commands".

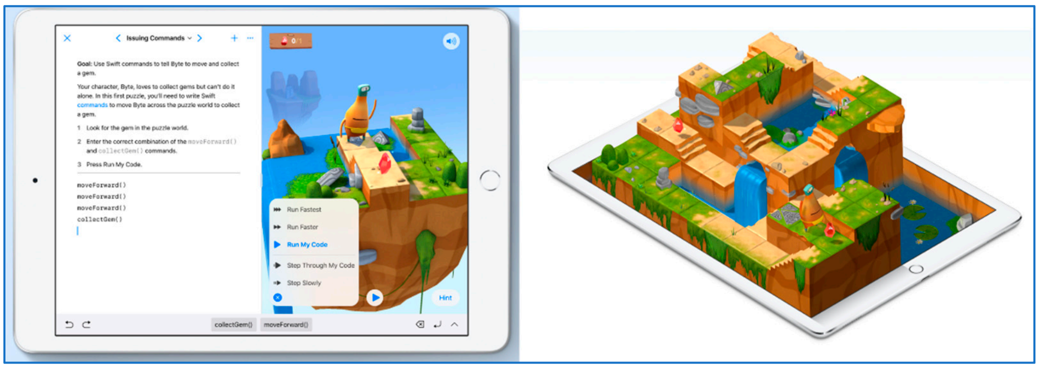


Figure 2. The Screenshot of the Swift Playground app presents the role of "Byte" and the scene of the task for the coding game (Retrieved 13 October 2020, from https://www.apple.com/swift/playgrounds, accessed on 13 October 2020).

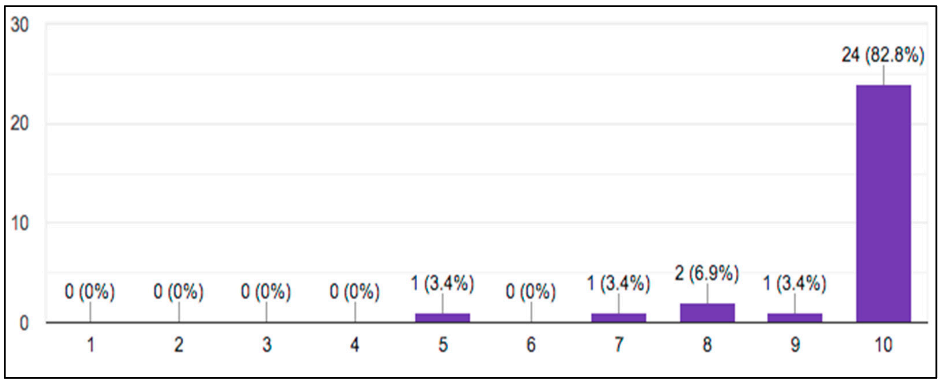


Figure 3. Difficulty analysis of Task I: "Issuing Commands".

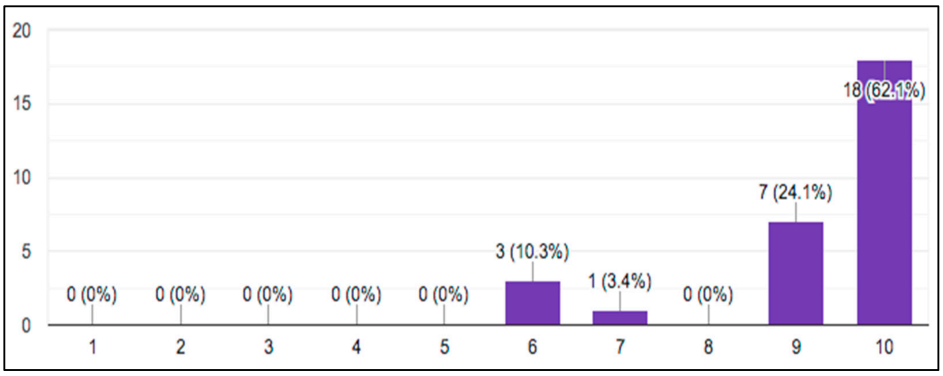


Figure 4. Difficulty analysis of Task II: "Adding a New Command".

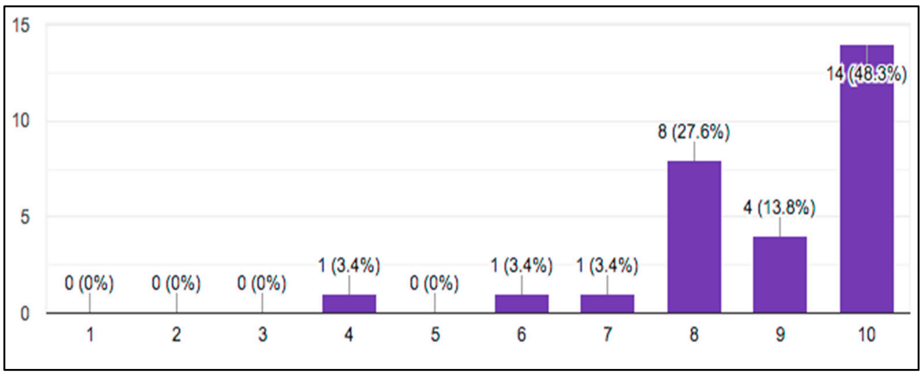


Figure 5. Difficulty analysis of Task III: "Toggling a Switch".

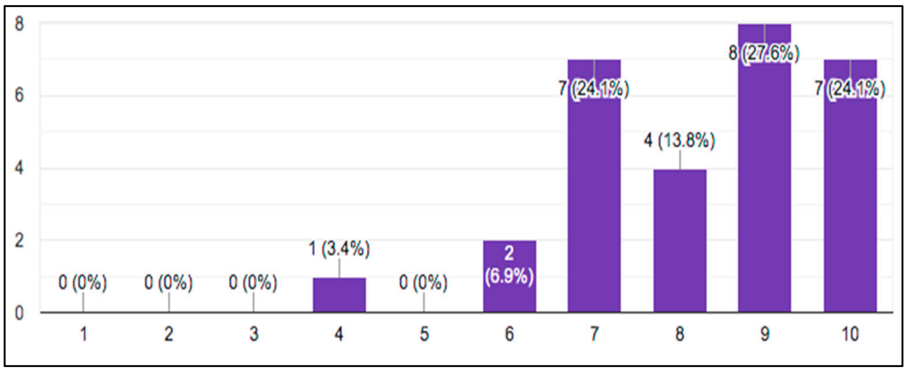


Figure 6. Difficulty analysis of Task IV: "Portal Practice".

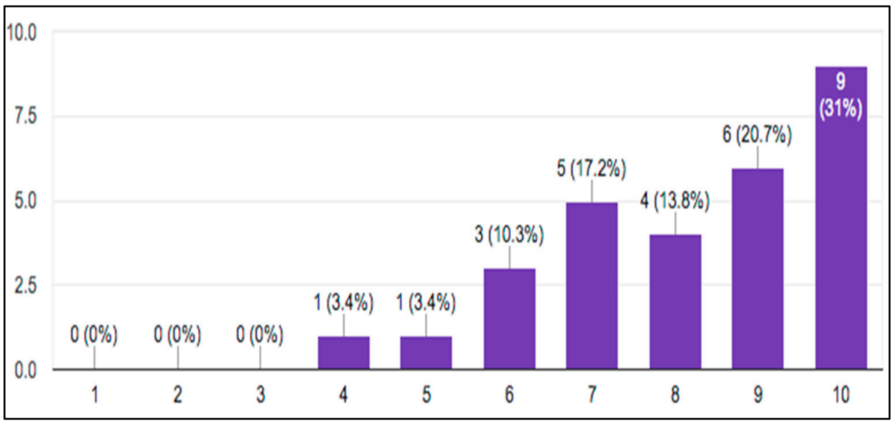
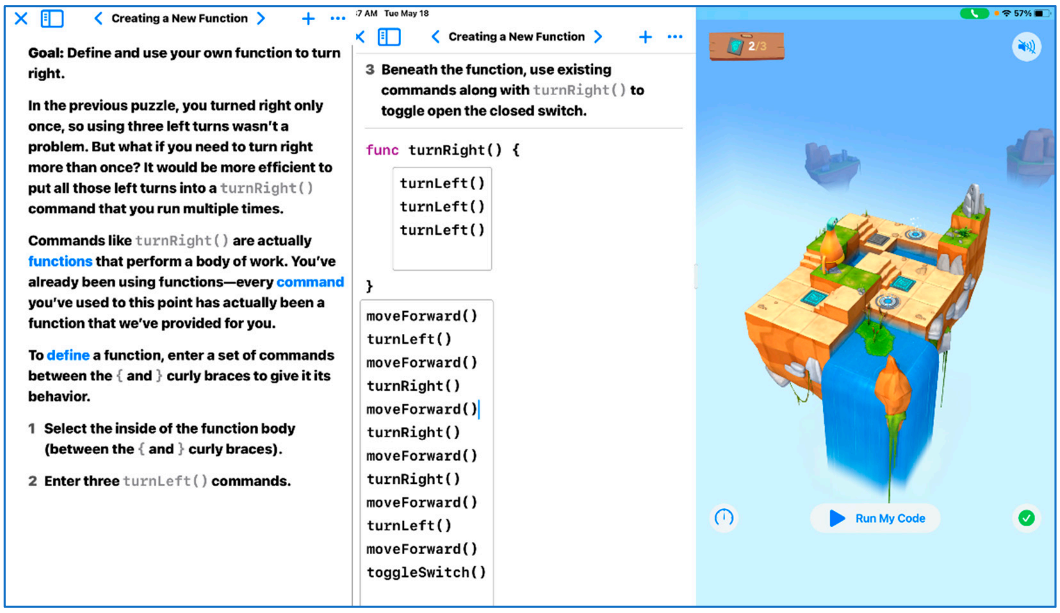


Figure 7. Difficulty analysis of Task V: "Composing a New Behavior".



Goal: Define and use your own function to turn right.

In the previous puzzle, you turned right only once, so using three left turns wasn't a problem. But what if you need to turn right more than once? It would be more efficient to put all those left turns into a turnRight() command that you run multiple times.

Commands like turnRight() are actually functions that perform a body of work. You've already been using functions—every command you've used to this point has actually been a function that we've provided for you.

To define a function, enter a set of commands between the { and } curly braces to give it its behavior.

1 Select the inside of the function body (between the { and } curly braces).

2 Enter three turnLeft() commands.

3 Beneath the function, use existing commands along with turnRight() to toggle open the closed switch.

```
func turnRight() {
    turnLeft()
    turnLeft()
    turnLeft()
}
moveForward()
turnLeft()
moveForward()
turnRight()
moveForward()
turnRight()
moveForward()
turnRight()
moveForward()
turnLeft()
moveForward()
toggleSwitch()
```

Figure 8. Task VI: "Creating a New Funtion".

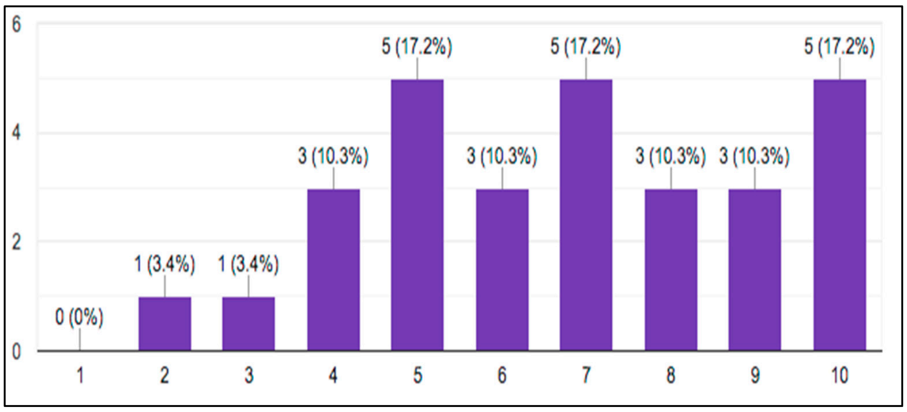
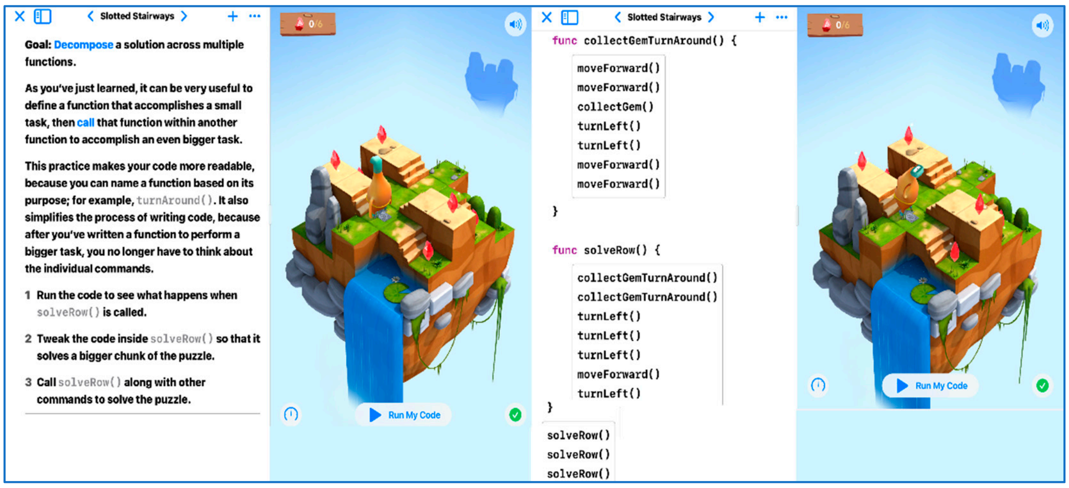


Figure 9. Difficulty analysis of Task VI: "Creating a New Funtion".



Goal: Decompose a solution across multiple functions.

As you've just learned, it can be very useful to define a function that accomplishes a small task, then call that function within another function to accomplish an even bigger task.

This practice makes your code more readable, because you can name a function based on its purpose; for example, `turnAround()`. It also simplifies the process of writing code, because after you've written a function to perform a bigger task, you no longer have to think about the individual commands.

1 Run the code to see what happens when `solveRow()` is called.

2 Tweak the code inside `solveRow()` so that it solves a bigger chunk of the puzzle.

3 Call `solveRow()` along with other commands to solve the puzzle.

```
func collectGemTurnAround() {
    moveForward()
    moveForward()
    collectGem()
    turnLeft()
    turnLeft()
    moveForward()
    moveForward()
}

func solveRow() {
    collectGemTurnAround()
    collectGemTurnAround()
    turnLeft()
    turnLeft()
    turnLeft()
    moveForward()
    turnLeft()
}
solveRow()
solveRow()
solveRow()
```

Figure 10. Task VII: "Slotted Stairways".

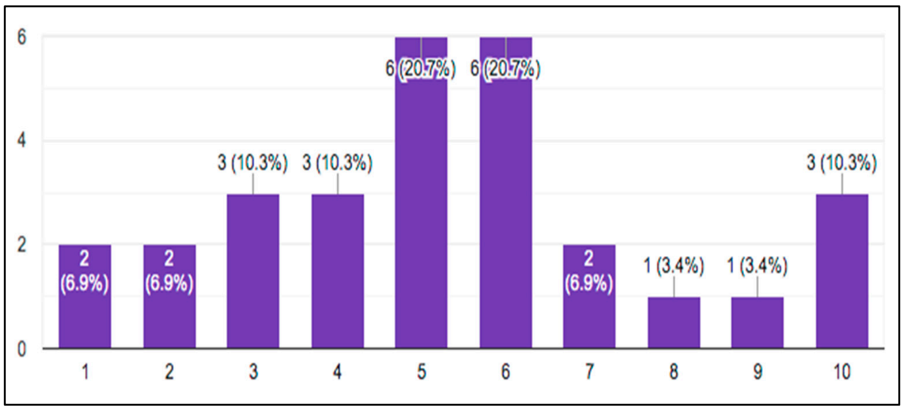
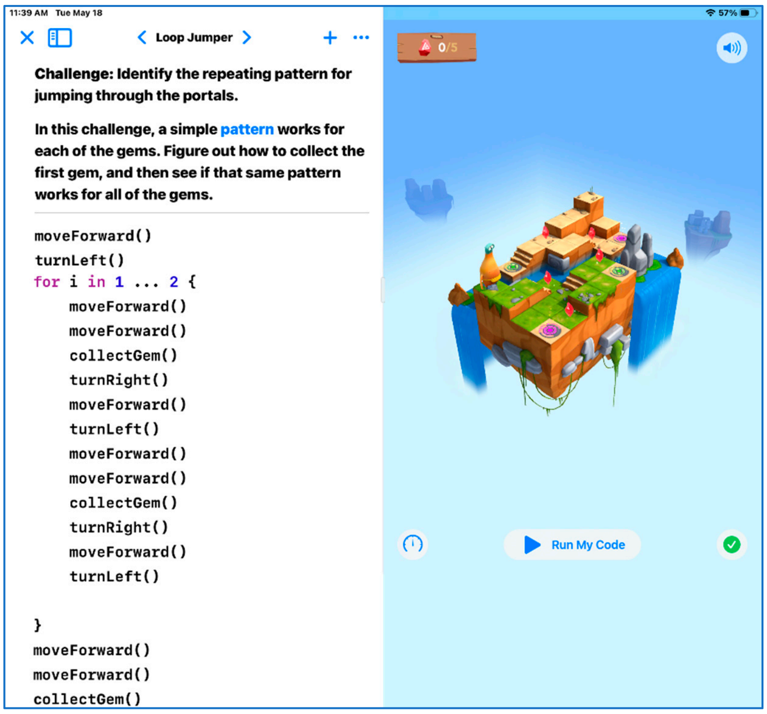


Figure 11. Difficulty analysis of Task VII: "Slotted Stairways".



Challenge: Identify the repeating pattern for jumping through the portals.

In this challenge, a simple pattern works for each of the gems. Figure out how to collect the first gem, and then see if that same pattern works for all of the gems.

```
moveForward()
turnLeft()
for i in 1 ... 2 {
    moveForward()
    moveForward()
    collectGem()
    turnRight()
    moveForward()
    turnLeft()
    moveForward()
    moveForward()
    collectGem()
    turnRight()
    moveForward()
    turnLeft()

}
moveForward()
moveForward()
collectGem()
```

Figure 12. Task VIII: "Loop Jumper".

3.1.1. Coding Command (80 min)

Task I: Preceding the "Issuing Commands" task in "Hello! Byte" on Swift Playgrounds, the teacher displays the iPad picture, prompts task goals, touches it with their finger, and writes to add commands moveForward() and collectGem(). After adding the commands and pressing "execute my code" on the picture, the "Byte" moves forward 3 steps (1 step for going up/down the stairs), collects jewels, and reaches the destination. In Figure 3, the result of the difficulty analysis for class students learning Task I: "Issuing Commands", 24 students, among the 29, considered the degree of ease to be 10 (82.8%), 1 student considered the degree of ease to be 9 (3.4%), and 2 students considered the degree of ease to be 8 (6.9%). In total, 27 students (93.1%) considered Task I: "Issuing Commands" to be easy.

Task II: Preceding the "Adding a New Command" task in "Hello! Byte" on Swift Playgrounds, the teacher demonstrates the iPad picture, prompts task goals, continues the previous task, and adds the command turnLeft(). After adding the command and pressing "execute my code" on the picture, the "Byte" moves forward 2 steps, turns left, moves forward 2 steps, and collects jewels to reach the destination. In Figure 4, the result of the difficulty analysis for class students learning Task II: "Adding a New Command", 18 students considered the degree of ease to be 10 (62.1%) and 7 students considered the degree of ease to be 9 (24.1%). In total, 25 students (86.2%) considered Task II: "Adding a New Command" to be easy.

Task III: Preceding the "Toggling a Switch" task in "Hello! Byte" on Swift Playgrounds, the teacher displays the iPad picture, prompts task goals, continues the previous task, and adds the command toggleSwitch(). After adding the command and pressing "execute my code" on the picture, the "Byte" moves forward 2 steps, turns left, moves forward, collects jewels, moves forward, turns left, moves forward, and performs a Toggling a Switch to reach the destination. In Figure 5, the result of the difficulty analysis for class students learning Task III: "Toggling a Switch", 14 students considered the degree of ease to be 10 (48.3%), 4 students considered the degree of ease to be 9 (13.8%), and 8 students considered the degree of ease to be 8 (27.6%). In total, 26 students (89.7%) considered Task III: "Toggling a Switch" to be easy.

Task IV: Preceding the "Portal Practice" task in "Hello! Byte" on Swift Playgrounds, the teacher demonstrates the iPad picture, prompts task goals, continues the previous task, and adds the command toggleSwitch(). After adding the command and pressing "execute my code" on the picture, the "Byte" moves forward 3 steps, turns left, moves forward 2 steps, does a Toggling a Switch, moves forward, enters the Portal, exits the Portal, moves forward, turns left, moves forward 2 steps, and collects jewels to reach the destination. In Figure 6, the result of the difficulty analysis for class students learning Task IV: "Portal Practice", 7 students considered the degree of ease to be 10 (24.1%), 8 students considered the degree of ease to be 9 (27.6%), and 4 students considered the degree of ease to be 4 (13.8%). In total, 19 students (65.5%) considered Task IV: "Portal Practice" to be easy.

3.1.2. Building Functions (80 min)

Task V: Preceding the "Composing a New Behavior" task in "Hello! Byte" on Swift Playgrounds, the teacher displays the iPad picture, prompts task goals, adds the following commands, and presses "execute my code" on the picture. The "Byte" moves forward 3 steps, turns left 3 times (without the command to turn right), moves forward 3 steps, and collects jewels to reach the destination. In Figure 7, the result of the difficulty analysis for class students learning Task V: "Composing a New Behavior", 9 students considered the degree of ease to be 10 (31%), 6 students considered the degree of ease to be 9 (20.7%), and 4 students considered the degree of ease to be 4 (13.8%). In total, 19 students (65.5%) considered Task V: "Composing a New Behavior" to be easy.

Task VI: Preceding the "Creating a New Funtion" task in "Hello! Byte" on Swift Playgrounds, the teacher demonstrates the iPad picture, prompts task goals, and establishes a turnRight() function by adding the command turnLeft() 3 times in func turnRight(){ }, and subsequently uses the function to complete the program command and function, as shown in Figure 8. By pressing "execute my code" on the picture, the "Byte" moves forward, turns left, moves forward, turns right, moves forward, turns right, moves forward, enters the Portal, exits the Portal, turns right, moves forward, turns left, moves forward, and does a Toggling a Switch to reach the destination. In Figure 9, the result of the difficulty analysis for class students learning Task VI: "Creating a New Funtion", 11 students (37.8%) considered Task VI to be easy (degree of ease 8–10), 16 students (55%) considered Task VI to be moderate (degree of ease 4–7), and 2 students (6.8%) considered Task VI to be difficult (degree of ease 1–3).

Task VII: Preceding the "Slotted Stairways" task in "Hello! Byte" on Swift Playgrounds, the teacher displays the iPad picture and prompts the task picture and goals as in the following figure. The "Byte" repeatedly collects jewels back and forth. This major task can be decomposed into 3 minor tasks, which are simplified with functions or commands for subsequent use of the Slotted Stairways. To practice the establishment of the collectGemTurnAround() function, in Figure 10, the commands to move forward 2 steps, collect jewels, turn left twice (turn backward), and move forward 2 steps are added in func collectGemTurnAround(){ }. To practice the establishment of the sloveRow(){} to complete the minor task of collecting 2 jewels, on the left of the figure, the commands collectGemTurnAround() 2 times, turn left 3 times (turning right), move forward, and turn left are added in func sloveRow(){ }. By pressing "execute my code" on the picture, the "Byte" executes the different commands, functions, and Slotted Stairways. In Figure 11, the result of the difficulty analysis for class students learning Task VII: "Slotted Stairways", 5 students (17.1%) considered Task VII to be easy (degree of ease 8–10), 17 students (58.6%) considered Task VII to be moderate (degree of ease 4–7), and 7 students (24.1%) considered Task VII to be difficult (degree of ease 1–3).

3.1.3. Building Loops (80 min)

Task VIII: Preceding the "Loop Jumper" task in "Hello! Byte" on Swift Playgrounds, the teacher demonstrates the iPad picture and prompts task pictures and goals. To find the step for repeatedly operating tasks on the picture, the part which is to be repeatedly executed can be searched on the task picture. After adding commands to move forward 2 steps, collect jewels, turn right, move forward, turn left, move forward 2 steps, collect jewels, turn right, move forward, enter Portal, exit Portal, and turn left for *i* in 1 . . . 2 { }, "execute my code" in the picture is pressed to have the "Byte" move forward, turn left, repeat the above loop twice, move forward 2 steps, and collect jewels to reach the destination, in Figure 12. In Figure 13, the result of the difficulty analysis for class students learning Task VIII: "Loop Jumper", 5 students (17.1%) considered Task VIII to be easy (degree of ease 8–10), 14 students (48.2%) considered Task VIII to be moderate (degree of ease 4–7), and 10 students (34.4%) considered Task VIII to be difficult (degree of ease 1–3).

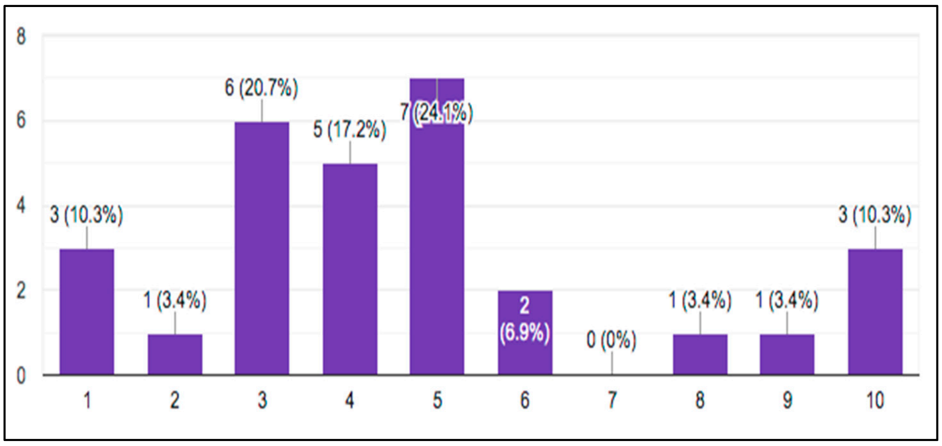


Figure 13. Difficulty analysis of Task VIII: "Loop Jumper".

The students are requested to fill in the Google form "student feedback on iPad Swift Playgrounds learning".

3.2. Qualitative Feedback Analysis after Teachers' Assessments of Swift Playgrounds

Nine teachers—2 gifted education program teachers, 2 English teachers, 2 ICT teachers, 2 science teachers, and 1 art teacher—were asked to assess the Swift Playgrounds iPad app. During the Professional Learning Community (PLC) gathering time, lasting about 2 h, they learned computational thinking and programming on their own and then filled in the Google form. From Table 1, which presents qualitative feedback analysis of the teachers' assessments of the computational thinking curriculum, most teachers considered that the basic course "Hello! Byte" on the Swift Playgrounds iPad app could train logical thinking and reasoning ability to largely help beginners learn a basic programming.

Table 1. Qualitative feedback analysis after teachers' assessments of Swift Playgrounds.

Teacher	Subject	The Most Impressive?	Importance of Programming to the Future?
Teacher A	gifted education program	The picture is exquisite and cute and the instruction steps are clear for self-learning.	Helps students dismantle problems, think of problem-solving steps, and write the steps with a specific execution sequence.
Teacher B	gifted education program	A fun program. Hopefully, I can continue to play and learn at home.	Everything could be controlled with programs. It allows me to realize that there are still many things to invent.
Teacher C	English	It is interesting to combine English with coding, and would be motivating for learning English.	Trains logic thinking, reasoning, computation, and, of course, English ability.
Teacher D	information technology education	It teaches programming with a drag-and-drop app and is presented with text, allowing students to get into the programming world earlier. It is impressive.	The app could train personal logic and allow oneself to better organize both programs and life. It is an excellent app.
Teacher E	science and technology	It combines complete space and logic concepts to easily attract learners practicing the course step by step. It is a good teaching material.	Nil
Teacher F	English	After comprehending the hierarchical learning process, it is easy to execute.	It allows for learning different languages and cultivating logic reasoning capability.
Teacher G	art and humanities	It is quite interesting and needs time for solving problems.	It could be combined with information technology to enhance students' learning interests.
Teacher H	information technology education	It has a rich picture/text interface to largely help beginners learn basic programming design.	Basic logic concept establishment and simple programming applications.
Teacher I	science and technology	It could train logical thinking and reasoning capability. Tasks are interesting.	Structured learning.

4. Conclusions and Suggestions

In the 12-year Basic Education practiced in 2019, the technology field reinforced problem solving and programming in computational thinking. This study re-wrote a lesson plan for technology pilot schools in the 2018 academic year into a paper. Computational thinking skills are becoming essential in all aspects of work and life and have become a part of the K-12 curriculum around the world [30]. For the many different program languages and computational thinking courses, the use of different training and learning tools has essential learning effectiveness [20,31–33]. In the study, a Swift Playgrounds computational thinking curriculum, lasting six sessions, was first developed, and nine elementary school teachers were asked to assess Swift Playgrounds. It was discovered that the tool could train students in logical thinking and reasoning capability. After the research, most teachers considered the tool as being able to train logical thinking and reasoning capability. Analysis of the students' learning feedback showed that 86% and 37% of students regarded adding commands and functions, respectively, as being easy, while 24% and 34% of students considered applying the unit step function and using loops, respectively, as being difficult. It is suggested that the curriculum should be explained in detail, or the schedule extended to allow most students to keep up with the schedule.

Before the end of the course, the teacher announced the codes for all tasks. This allowed students to build the learning scaffold and complete task operations more fluently during self-learning. All students were asked to fill in their feedback on a Google form in the last session, for summative evaluation. Swift Playgrounds is an iOS app. It can only be learned on an iPad, and most schools in the nation could not furnish each student, or even each class, with an iPad for this learning experience. A class was therefore arranged for trial teaching in this study. For a second class, we would need to establish students in different classes but with the same seat number on Swift Playgrounds for the "Hello! Byte" course. These restrictions might be factors that adversely affect the popularity of the course. Apple could release the app for different platforms to allow access to more teachers and students for learning.

Author Contributions: Data curation, G.-M.C.; validation, C.-P.C.; writing—original draft, G.-M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Helsinki, and approved by the Institutional Review Board.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

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

References

1. Joly, M.; Rondó, P.H. The future of computational biomedicine: Complex systems thinking. *Math. Comput. Simul.* **2017**, *132*, 1–27. [CrossRef]
2. Tikva, C.; Tambouris, E. Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Comput. Educ.* **2021**, *162*, 104083. [CrossRef]
3. Grover, S.; Pea, R. Computational thinking in K–12: A review of the state of the field. *Educ. Res.* **2013**, *42*, 38–43. [CrossRef]
4. Lye, S.Y.; Koh, J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Comput. Hum. Behav.* **2014**, *41*, 51–61. [CrossRef]
5. Carlborg, N.; Tyrén, M.; Heath, C.; Eriksson, E. The scope of autonomy when teaching computational thinking in primary school. *Int. J. Child-Comput. Interact.* **2019**, *21*, 130–139. [CrossRef]
6. Durak, H.Y.; Saritepeci, M. Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Comput. Educ.* **2018**, *116*, 191–202. [CrossRef]
7. Fidai, A.; Capraro, M.M.; Capraro, R.M. "Scratch"-ing computational thinking with Arduino: A meta-analysis. *Think. Ski. Creat.* **2020**, *38*, 100726. [CrossRef]
8. Bhatt, A.J.; Gupta, C.; Mittal, S. iABC: Towards a hybrid framework for analyzing and classifying behaviour of iOS applications using static and dynamic analysis. *J. Inf. Secur. Appl.* **2018**, *41*, 144–158. [CrossRef]
9. Seliverstov, Y.; Starichenkov, A.; Nikitin, K. Using mobile applications to evaluate quality of road networks and transport mobility. *Transp. Res. Procedia* **2020**, *50*, 636–646. [CrossRef]
10. Hanson, J. Not Just Another Coding App. *Sch. Libr. J.* **2016**, *62*, 18–20.
11. Hou, H.-Y.; Agrawal, S.; Lee, C.-F. Computational thinking training with technology for non-information undergraduates. *Think. Ski. Creat.* **2020**, *38*, 100720. [CrossRef]
12. Lin, S.-Y.; Chien, S.-Y.; Hsiao, C.-L.; Hsia, C.-H.; Chao, K.-M. Enhancing Computational Thinking Capability of Preschool Children by Game-based Smart Toys. *Electron. Commer. Res. Appl.* **2020**, *44*, 101011. [CrossRef]
13. Zhang, L.; Nouri, J. A systematic review of learning computational thinking through Scratch in K-9. *Comput. Educ.* **2019**, *141*, 103607. [CrossRef]

14. Dagienė, V.; Sentance, S.; Stupurienė, G. Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica* **2017**, *28*, 23–44. [CrossRef]
15. Taiwan ministry of Education. *Curriculum Guidelines of 12-Year Basic Education for Elementary, Junior High Schools and General Senior High Schools-Technology*; Taiwan ministry of Education: Taipei, Taiwan, 2020.
16. Wing, J.M. Computational thinking. *Commun. ACM* **2006**, *49*, 33–35. [CrossRef]
17. Wing, J.M. Computational thinking and thinking about computing. *Philos. Trans. R. Soc. Lond. A Math. Phys. Eng. Sci.* **2008**, *366*, 3717–3725.
18. Rowe, E.; Almeda, M.V.; Asbell-Clarke, J.; Scruggs, R.; Baker, R.; Bardar, E.; Gasca, S. Assessing implicit computational thinking in Zoombinis puzzle gameplay. *Comput. Hum. Behav.* **2021**, *120*, 106707. [CrossRef]
19. Kert, S.B.; Erkoç, M.F.; Yeni, S. The effect of robotics on six graders' academic achievement, computational thinking skills and conceptual knowledge levels. *Think. Skills Creat.* **2020**, *38*, 100714. [CrossRef]
20. Chen, G.; Shen, J.; Barth-Cohen, L.; Jiang, S.; Huang, X.; Eltoukhy, M. Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Comput. Educ.* **2017**, *109*, 162–175. [CrossRef]
21. Heintz, F.; Mannila, L. Computational thinking for all: An experience report on scaling up teaching computational thinking to all students in a major city in Sweden. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, MD, USA, 21–24 February 2018; ACM: Mew York, NY, USA, 2018; pp. 137–142.
22. Bellettini, C.; Carimati, F.; Lonati, V.; Macoratti, R.; Malchiodi, D.; Monga, M.; Morpurgo, A. A Platform for the Italian Bebras. In Proceedings of the 2018 International Conference on Computer Supported Education, Madeira, Portugal, 15–17 March 2018; pp. 350–357.
23. Izu, C.; Mirolo, C.; Settle, A.; Mannila, L.; Stupuriene, G. Exploring Bebras Tasks Content and Performance: A Multinational Study. *Inform. Educ.* **2017**, *16*, 39–59. [CrossRef]
24. Grossman, M.; Aziz, M.; Chi, H.; Tibrewal, A.; Imam, S.; Sarkar, V. Pedagogy and tools for teaching parallel computing at the sophomore undergraduate level. *J. Parallel Distrib. Comput.* **2017**, *105*, 18–30. [CrossRef]
25. Chen, C.P. Using Padlet cooperative learning for computational thinking challenge in elementary schools. In Proceedings of the TANET 2018 Taiwan Academic Network Conference, Taoyuan, Taiwan, 24–26 October 2018.
26. Coding Clubs. Everyone Can Code Chicago. Available online: https://www.eccchicago.org/everyone-can-code-chicago-coding-clubs.html (accessed on 7 July 2017).
27. Bers, M.U.; González-González, C.; Armas-Torres, M.B. Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Comput. Educ.* **2019**, *138*, 130–145. [CrossRef]
28. Falloon, G. An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *J. Comput. Assist. Learn.* **2016**, *32*, 576–593. [CrossRef]
29. Bruni, F.; Onofrio, L.; Nisdeo, M. Start App: A coding experience between primary and secondary school. *Form@re Open J. Form. Rete* **2016**, *16*, 188–200.
30. Wei, X.; Lin, L.; Meng, N.; Tan, W.; Kong, S.-C. The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. *Comput. Educ.* **2021**, *160*, 104023. [CrossRef]
31. Asbell-Clarke, J.; Rowe, E.; Almeda, V.; Edwards, T.; Bardar, E.; Gasca, S.; Baker, R.; Scruggs, R. The development of students' computational thinking practices in elementary-and middle-school classes using the learning game, Zoombinis. *Comput. Hum. Behav.* **2021**, *115*, 106587. [CrossRef]
32. Del Olmo-Muñoz, J.; Cózar-Gutiérrez, R.; González-Calero, J.A. Computational thinking through unplugged activities in early years of Primary Education. *Comput. Educ.* **2020**, *150*, 103832. [CrossRef]
33. Lei, H.; Chiu, M.M.; Li, F.; Wang, X.; Geng, Y.-J. Computational thinking and academic achievement: A meta-analysis among students. *Child. Youth Serv. Rev.* **2020**, *118*, 105439. [CrossRef]