International Society of the Learning Sciences

Parsing the Use of Computational Concepts with Scratch Projects

Joey Huang, North Carolina State University, joeyhuang@ncsu.edu

Abstract: Studies in learning sciences have examined the learning of computational thinking through project-based learning. The application of computational concepts has been proven to be related to specific project types. Although studies have suggested the value of examining the relationship between computational concepts and project type, few have addressed the nuances regarding how different project types support specific concepts, as well as how educators could use to inform instructional design. This study examined students' group projects to understand how computational concepts associate with specific project types. The implications of the findings inform how educators can better design the instructional and learning objectives to facilitate computational thinking through project-based learning.

Introduction

In recent years, extensive research has focused on teaching and learning through project-based learning (PBL) and computational thinking (CT) (Saad & Zainudin, 2022). Studies have explored PBL's application across diverse domains such as robotics, big data, game design, and programming (Chiu, 2020). Notably, there has been widespread adoption of PBL within computer science courses, which has focused on CT as a central concept in teaching and learning. Scratch is a prevalent tool to teach CT with PBL approach (Zhang & Nouri, 2019). Additionally, project types (e.g., games, animations, story) have been proven to demonstrate different uses of CT concepts (Park & Shin, 2019). Previous studies have found that games and animation support specific CT concepts, such as loops, user interaction for younger students (Maloney et al., 2008). Particularly, games projects have a higher number of uses on CT concepts like parallelism, conditionals, or data than projects like music or stories. Although studies have suggested the value of examining the relationship between CT concepts and project type, few studies addressed the nuances of how different project types support different CT concepts and how educators might use this knowledge in instructional design. Understanding how these CT concepts are associated with specific project types will help advance instructional design and learning objectives. It can provide a more holistic view by applying project design to facilitate the learning of CT concepts. In this study, I ask: 1. To what extent do students in small groups apply CT concepts differently across project types? 2. How do student groups utilize CT concepts across different types of projects?

Methods

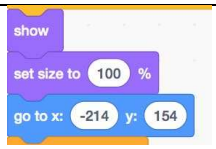
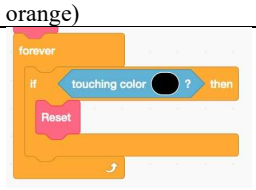
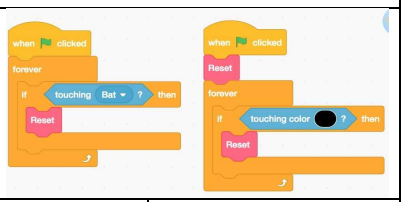
The data were collected in a public middle school in a midwestern U.S. state that developed K-12 CS curriculum standards. They were collected as part of a five-week curriculum in an Introduction to Computer Science class. The data included four triad groups with a total of 12 students (Female = 4, Male = 8). The groups consisted of a mixture of novice to experienced students. The majority of students were Caucasian (n=8), two were Asian, and two were Hispanic. During the five-week curriculum, students worked in small groups to complete different types of projects. A total of 25 final projects (five from each group per project type) were collected and categorized into five project types - music, animation, interactive collage, story, and games. The projects were coded based on the seven concepts from Brennan & Resnick (2012)'s framework: sequences, loops, parallelisms, events, conditionals, operators, and data (see Table 1). Collaborating with a trained researcher familiar with Scratch, we analyzed the group projects together. We coded a total of 25 projects together. The group projects were exported as JSON files to examine the number of use of events (yellow blocks, i.e., when green flag, when this sprite clicked, broadcast message one), loops (repeat or forever blocks), conditionals (orange blocks, i.e., if-then, wait) (because the conditionals overlap with repeat or forever blocks, the analysis only included the ones outside these two blocks as conditionals), operators (green blocks), and data (variables blocks). In addition to these five CT concepts, we identified the use of sequences and parallelism by looking at the script of Scratch projects. Particularly, to identify as sequences, the blocks should include at least one event and at least two blocks after the event block, and parallelism indicates there were at least two identical triggers (i.e., when green flag clicked, when space key pressed, when this sprite clicked, when backdrop switches to x, when x > "variable", or when I receive message 1) in one sprite. Descriptive statistics were calculated, and the one-way ANOVA was applied to determine differences in each of the seven CT concept across five design projects.

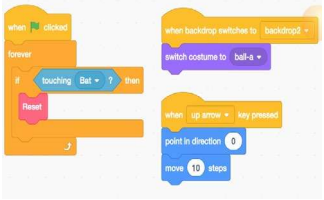
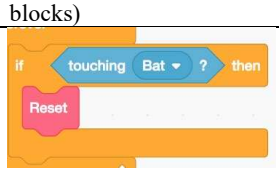
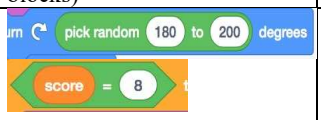
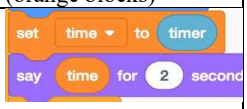
To understand how students applied CT concepts differently across project types, I detailed the post-hoc findings by providing a sample project of games to describe the use of the four CT concepts. Because these five project types were introduced in sequence during the implementation, I then examined the purpose and structure

of the blocks to understand if changes in students' use of CT concepts could be attributed solely to learning from the projects, or if the project type also had an impact.

Table 1

Computational Thinking Concepts (Brennan & Resnick, 2012)

Concept	Sequence	Loops	Parallelism
Description	Identifying a series of steps for a task	Running the same sequence multiple times (in light orange)	Making things happen at the same time
Example from Student Projects			

Events	Conditionals	Operators	Data
One thing causing another thing to happen (yellow blocks)	Making decisions based on conditions (light orange blocks)	Support for mathematical and logical expressions (green blocks)	Storing, retrieving, and updating values (orange blocks)
			

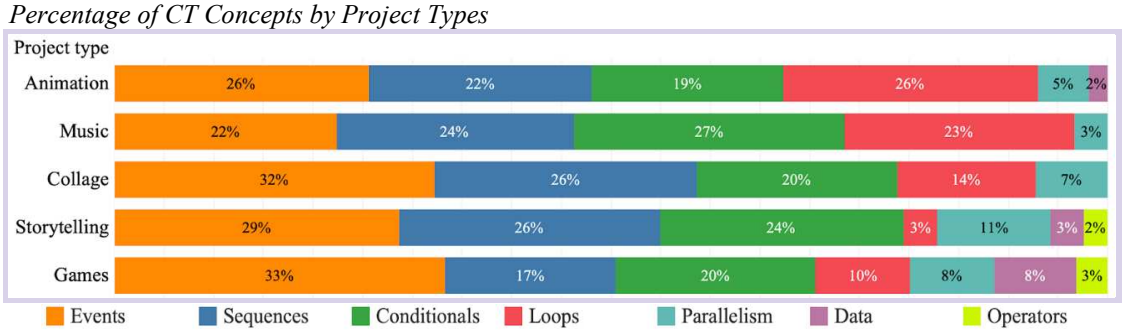
Results

Findings of computational concepts by project type

The results were presented as percentages to show the proportion of each concept applied in each of the five projects. Figure 1 illustrates that events, sequences, conditionals were the dominant concepts across five projects (between 17% - 33%). Particularly, these three concepts were emphasized as part of the instructional objectives in animation projects, which was the first group project. In addition to these three concepts, loops were included in the instructional objectives in the first three project units – animation, music, and collage. The groups applied higher percentages of loops on animation (23%) and music projects (26%) compared to the other three project types (less than 14%). Parallelism was applied between 3% to 11% across five project types. Data and operators were considered as more advanced concepts (Weintrop et al., 2018) compared to the other five concepts and were introduced later in the project units – story and games. However, a small percentage (2%) of data concept was used in couple animation projects, which included score variables as an interactive element between users and the program. The findings provided a general overview of the CT concepts applied across the five project types.

Figure 1

Percentage of CT Concepts by Project Types



In addition to analyzing the percentage of CT concepts by project type, I conducted a one-way ANOVA with frequency count to examine the differences in CT concepts across project types. Statistically significant differences were observed in four out of seven CT concepts, as determined by the one-way ANOVA. First, sequences ($F(4,20) = 3.338$, $p = .030$, $\eta^2 = .400$). A Tukey post-hoc test revealed that the CT concept, sequences,

was statistically significantly different for the project type between games and the other two project types (i.e., animation and music). The effect size is large. Next, a Tukey post-hoc test showed that the CT concept, events, was statistically significantly different for the project type between games and other four project types: music, collage, animation, and story with a large effect size. Events ($F(4,20) = 9.118$, $p = 0.000$, $\eta^2 = .646$). In addition to sequences and events, parallelism was statistically significantly different for the project type between games and other three project types: animation, music, collage, with a large effect size. Parallelism ($F(4,20) = 8.004$, $p = 0.001$, $\eta^2 = .616$). Finally, the post-hoc test revealed that the CT concept – data, was statistically significantly different for the project type between games and other four project types: music, collage, animation, and story and with a large effect size. Data ($F(4, 20) = 63.154$, $p = 0.000$, $\eta^2 = .927$). The results of the one-way ANOVA showed there were statistically significant differences between CT concepts and project types on four out of seven CT concepts: sequences, events, parallelism, and data. Additionally, the Tukey post-hoc tests showed that the games project was significantly different from the other four types of projects: music, collage, animation, and story on the CT concepts – data and events. Furthermore, the games project was significantly different from the three types of projects: animation, music, collage on the CT concept – parallelism. Finally, the games project was significantly different from the two types of projects: animation and music on sequences.

Sequences, events, data, and parallelism in games project

Building on the quantitative results, I then examined the use of these four CT concepts (i.e., sequences, events, parallelism, and data) in games. Due to the limited space, I provided an example from a maze game, created by a student group, to illustrate how students applied the concepts. The group designed a maze game with a goal of opening a treasure box by passing two levels of the maze, where the player needs to pass the first level to receive the key for the treasure box. The group included various obstacles, such as a bat in Figure 2.1, in the game. The group used a variety of sprites to complete the game design. Events were required for every sprite to trigger the following actions (events - yellow blocks). Additionally, to ensure the game ran successfully, the group needed to consider the series of actions of each sprite (sequences), and these actions had to run simultaneously (parallelism) for the game to function. Due to the complexity of the blocks, the group applied the data function (pink block). The use of data blocks increased efficiency for reviewing and modularization for further adaptation and script changes.

Figure 2.1

The Use of Events, Sequences, Parallelism, and Data in a Maze Game

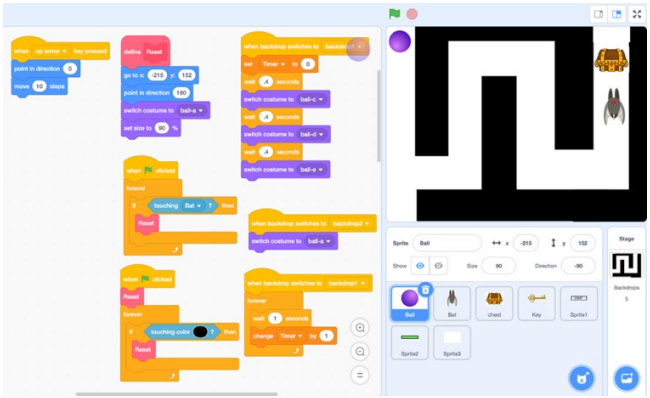
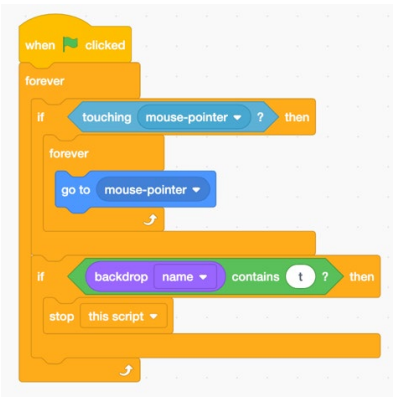


Figure 2.2

Use of Loops and Conditionals in a Maze Game



Loops were a key learning objective across most units, prominently used in animation, music, and collage projects to repeat actions like costume changes and movements. In contrast, their application was less frequent and more complex in story and games projects. Particularly, loops in games included a conditional statement to specify a condition to stop the iteration. Figure 2.2 shows the group applied multiple conditional blocks, “if-then,” which are nested in a forever block, and another forever block was applied inside the if-then conditional block. This usage of loops was common in games projects. Loops is a challenging concept to teach in programming, especially for novice learners (Grover et al., 2016). In animation, music, and collage projects, groups predominantly used "wait" and "wait until" blocks, while "if-then" conditional blocks were more common in story and games projects. Games projects also incorporated "clone" blocks, enabling sprites to duplicate themselves during runtime. Although not a specified instructional goal, this advanced use of "clone" and "if-then" blocks showcases the groups' adept handling of complex conditional logic. This reflects the tailored application of loops

and conditionals across different projects, with games requiring a deeper grasp of these concepts for effective script execution (Park & Shin, 2019). In addition to loops and conditionals, parallelism requires the groups to orchestrate simultaneous events that trigger multiple executions of programs. For animation and music projects, the groups often used event blocks, such as "when green flag clicked," "when the sprite is clicked," or "when (left arrow) key pressed" for multiple sprites to achieve the goals of the projects. The groups applied parallelism, which consisted of multiple different types of events, such as "broadcast" and "when backdrop switches to (backdrop 1)." These events were applied under multiple sprites in collage, story, and games projects. Particularly, for games projects, many programs of sprites needed to proceed simultaneously to achieve the requirements of the project.

Discussion

Building on previous studies, this study examines the nuances among CT concepts and their association with different project types. The results showed the distribution of the seven CT concepts across five project types — animation, music, collage, story, and games. Findings suggest that animation projects support the use of CT concepts like events and loops, while music projects support the use of conditionals (e.g., if-then, stop all). Collage and story projects demonstrated a higher number of sequences and events. Games projects featured the highest number of events among all project types, with conditionals and sequences being more frequent compared to the other four project types. Particularly, post-hoc results of the one-way ANOVA indicated a significant presence of four CT concepts: sequences, events, parallelism, and data.

The findings show that the complexity of projects increased progressively from the first unit (animation) to the last unit (games). Although it was not surprising to find a general progression in project complexity over time, certain CT concepts seemed to be associated with specific project types. In games projects, groups frequently applied conditional blocks nested with forever or repeat blocks (loops). Furthermore, the findings provide implications for instructional design. PBL has been widely employed in both formal and informal learning settings to teach CT (Lye & Koh, 2014). Tools like Scratch are designed with a "low floor" (easy to get started) and a "high ceiling" (complex projects) for students to engage their interests and create personal or group-meaningful projects. By understanding the use of CT concepts across project types, educators can better design the instructional and learning objectives to facilitate student learning. The findings provide insights for educators to develop and refine their project guidelines and curriculum to align instructional expectations and learning outcomes. Educators can consider project types to guide students in advancing specific CT concepts and refining their instructional goals.

References

Brennan, K., & Resnick, M. (2012, April). 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, Canada* (Vol. 1, p. 25).

Chiu, C. F. (2020). Facilitating k-12 teachers in creating apps by visual programming and project-based learning. *International Journal of Emerging Technologies in Learning (iJET)*, *15*(1), 103-118.

Grover, S., Pea, R., & Cooper, S. (2016, February). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM technical symposium on computing science education* (pp. 552-557).

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, *41*, 51-61.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008, March). Programming by choice: urban youth learning programming with scratch. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education* (pp. 367-371).

Park, Y., & Shin, Y. (2019). Comparing the effectiveness of scratch and app inventor with regard to learning computational thinking concepts. *Electronics*, *8*(11), 1269.

Saad, A., & Zainudin, S. (2022). A review of Project-Based Learning (PBL) and Computational Thinking (CT) in teaching and learning. *Learning and Motivation*, *78*, 101802.

Weintrop, D., Hansen, A. K., Harlow, D. B., & Franklin, D. (2018, August). Starting from Scratch: Outcomes of early computer science learning experiences and implications for what comes next. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (pp. 142-150).

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. Computers & Education, 141, 103607.