WearableLearning: Developing Computational Thinking Through Modeling, Simulation, and Computational Problem Solving

Injila Rasul, Danielle Crabtree, Francisco Castro, Allison Poh, Sai Satish Gattupalli, Krishna Chaitanya Rao Kathala, Ivon Arroyo
irasul@umass.edu, dcrabtree@umass.edu, fcastro@cs.umass.edu, apoh@umass.edu, sgattupalli@umass.edu, kkathala@umass.edu, ivon@cs.umass.edu
University of Massachusetts Amherst

Abstract: Computational Thinking (CT) is a vital and multi-dimensional skill for all learners. Understanding the development of CT's different dimensions is essential to refining educational experiences that best support it. In this study, we investigated the development of three aspects of CT: Self-Perception of Computational Ability, Modeling and Simulation, and Computational Problem Solving, as students engaged in collaborative game design and programming practices within a game-design platform, *WearableLearning* (WL). Through their engagement with WL and its accompanying curriculum, we show preliminary evidence for developing two CT dimensions, Modeling and Simulation and Computational Problem Solving, and discuss the challenge of developing Self-Perception of Computational Ability. We found an increase in students' ability to understand machines and their processes and an improved capacity to think algorithmically as they constructed models, debugged, and iterated through their designs. Student Self-Perceptions of Computational Ability, however, did not change significantly.

Introduction

Educators recognize that Computational thinking (CT) is an essential skill to master within STEM fields (Wing, 2006). Considered a key aspect of Computer Science (Grover & Pea, 2018), CT is now understood to more broadly encompass the thought processes and practices necessary for systematic problem-solving, such as iterative testing, algorithmic thinking, and troubleshooting, among others (Weintrop et al., 2016; Shute et al., 2017; Resnick et al., 2009). In particular, there is limited work that explores CT within environments that leverage game design to teach critical CT skills and concepts. Our work describes the results of engaging students in a six-stage curriculum (the WearableLearning curriculum) that teaches CT through programming and playing games on a web-based game design platform.

The *WearableLearning* Platform (WL) aims to develop students' CT skills by having K-12 students and teachers create, administer, and play multiplayer games using mobile devices (Arroyo et al., 2022). In this work, we sought to understand the impact of the WL curriculum that involves gameplay, game design, programming, and testing. We thus explore the following research question in this work: *what aspects of Computational Thinking does the process of game playing, game ideation, and programming/implementation impact*? We hypothesized that the aforementioned processes would yield an increase in students' CT abilities, in particular: how students see their abilities to solve problems (self-perception of computational ability), their understanding of machines and models that represent their functioning (modeling and simulation), and their capability to generate correct solutions that use logical reasoning (computational problem solving).

Background

Computational Thinking has become an umbrella term that refers to a broad set of early Computer Science skills essential to thrive in our increasingly digital world (Wing, 2006). Many scholars have grappled with the different definitions of CT and have contributed different, highly context-driven answers. Most definitions include the aspects of problem decomposition, pattern recognition, data representation, generalization and abstraction, systems thinking, and algorithm-building (Grover & Pea, 2018). Another one of the most thorough analyses of CT at the K-12 level (Weintrop et al., 2016) suggested the following practices within CT: analyzing and logically organizing data; data modeling, abstractions, and simulations; formulating problems so computers may assist; identifying, testing, and implementing possible solutions; automating solutions via algorithmic thinking; and generalizing this process to other problems. We agree with the reflection by Román-González et al. (2019), which suggests that the term *Computational Thinking* has helped to extend Computer Science Education beyond computer programming, and helped to lower the barriers to entry for learning computer programming, in part, due to an increase in the number of visual block-based programming languages. In addition to developing cognitive skills, engaging in CT offers the opportunity to develop noncognitive aspects related to attitudes and 21st-century skills such as persistence, self-confidence, tolerance to ambiguity, creativity, and teamwork (van Laar et al., 2017).

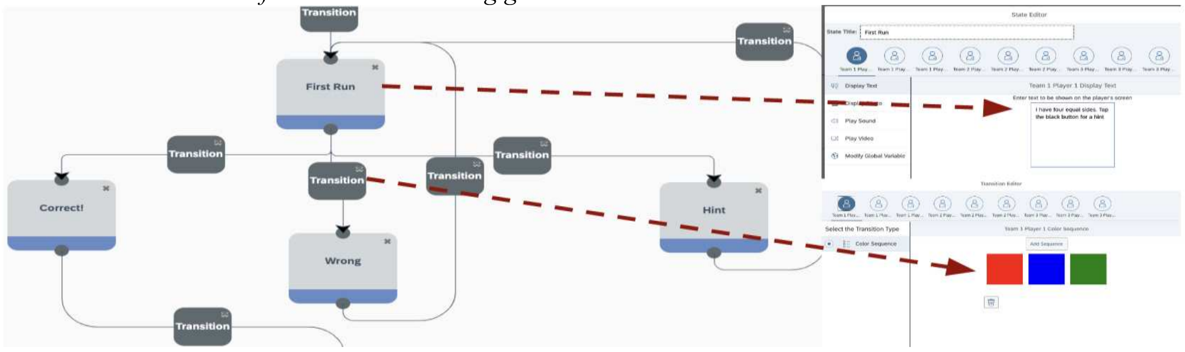
Thus, the uses and applications of CT have evolved and grown beyond strictly CS education (Kalelioglu et al., 2016).

Exploring the different dimensions of CT enables us to identify distinct mental processes during problem-solving tasks in contexts within and beyond coding and programming. Game design emerges as a context that is rich with opportunity to develop CT because students are tasked with creating a sequence of interfaces while keeping scale, difficulty, individual differentiation, and complexity in mind (Kafai et al., 2015). Engaging in CT often results in a logic-based computational solution to a problem that is defined at various levels of abstraction. CT involves a cyclical process where a problem is explored at a high level of detail and precision, with multiple possible solutions. These solutions need to be articulated and implemented to varying degrees, tested according to a success criterion, and revised or redefined.

The WearableLearning platform

The WL Platform is a browser-based platform that enables users to create and deploy active math games without prior programming experience (Castro, 2022). The programming interface (see Figure 1) consists of *states* and *transitions* that users drag and drop onto a canvas on the screen to create a game flow. *States* contain specific game content that are displayed to a game player, such as text (e.g., questions, prompts), images, sound, and videos. Movement from one state to the next depends on the *transitions* that connect states. A player can enter text, color codes, or press buttons corresponding to choices to transition from one state to the next. WL's Game Creator functionality includes a debugger, which runs a simulated game instance to show what a Game Player would see on their screen as they progress through the game, allowing users to troubleshoot, debug, and fix errors as they develop their game.

Figure 1
The Game Editor view of a WearableLearning game.



I have four equal sides. Tap the black button for a hint.

🟥 🟦 🟩

The WearableLearning curriculum

The WL curriculum consists of a series of guidelines and materials to support teachers in guiding their students through a six-stage process of playing, creating, and modifying games. The overarching goal is to design an active, educational, collaborative math game aided by mobile devices for learners to play, learn, and practice math skills. Students work through an iterative design process and engage in designing, prototyping, and testing/evaluating their game designs, and redesigning after feedback from other students. Students follow a six-stage design process: (1) playing games, (2) brainstorming and designing a game on paper, (3) drawing finite state machine diagrams, (4) programming the games on the WL platform, (5) playing a math game made by peers, and (6) observing others play their math game and iterating on their game based on these observations.

Methodology

To address our research question, we conducted a study with 47 students (11–13 years old), across two after-school programs in Eastern and Western Massachusetts in the United States, where students experienced the six-stage WL curriculum over 8-10 hours of contact time. Across both programs, 22 (46.8%) of students self-identified as male, 18 (38.3%) self-identified as female, and 7 (14.9%) preferred not to answer.

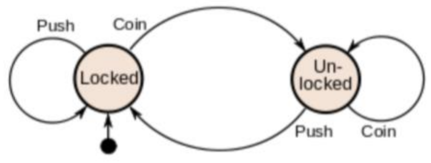
Instruments

We designed pre- and post-tests to measure three different aspects of CT: Self-Perception of Ability, Modeling and Simulation, and Computational Problem Solving, before and after students engaged in the WL curriculum. The full curriculum and instruments are at https://osf.io/tx9ab/. The first section of each test gleaned students' self-concept of how adept they considered themselves at computational tasks and the use of computers. This section was adapted from Angeli et al.'s (2016) work on CT perspectives where they describe what students at different grade levels should know to have CT. We created a test of CT self-concept by adapting these notions to feelings of knowing CT skills into seven 5-point Likert scale questions (5="Strongly Agree" to 1="Strongly Disagree").

The second part of the test posed questions related to modeling a machine (Figure 2 shows sample items on a finite state machines assessment). According to Wilensky et al. (2016), CT activities in various capacities include learners designing, constructing, and evaluating models as part of their educational activities. We thus assessed students' modeling and simulation ability by assessing their ability to interpret abstract models of finite state machines, what kinds of machines they represent, and how they function, especially given the heavy focus of the WL curriculum and platform on finite-state machines. We created items on Finite State Machines from beginner material given to students in tertiary education as part of Computer Science classes.

Figure 2
The three items that assessed students' knowledge of Finite State Machines

This is a visual model of how a machine works.



1. Can you tell what kind of machine this could be? What does it remind you of and why?
2. According to this model, what would happen if you put three (3) coins in?
3. The machine is currently locked. What would you have to do to get this machine to go "unlocked" and then back to "locked"?

The final part of the test had multiple-choice items from the Computational Thinking test (CTt) by Roman Gonzalez et al. (2019) relevant to algorithmic thinking concepts, such as sequencing and loops, to measure computational problem-solving skills. According to Wilensky (2016), this involves a variety of skills, such as preparing problems for computational solutions, choosing tools, assessing approaches/solutions to a problem, and debugging. The CTt addresses some of these aspects; we adapted from the CTt and included only the preliminary concepts of algorithmic thinking, excluding the more complex nested loops and conditionals, as those were not part of the topics covered in the instruction for the after-school programs.

Results

We computed a total score for each student for the pre-/post-tests, with the score for self-perception as an average of the Likert scale scores for each student and the FSM and CTt components being sums to indicate correctness. We computed descriptive statistics for the collected data and analyzed the difference between pre- and post-test scores using a two-tailed paired-samples t-test (significance compared to an alpha level of 0.05).

We found a statistical significance in the CT measures for Modeling & Simulation (t=2.01, p=0.05) and Computational Problem Solving (t=2.62, p=0.01) from pre to post-test. Modeling & Simulation showed a small to medium effect size, suggesting that exposure to the WL curriculum activities positively impacted students' skills in creating models and understanding finite state machine concepts. The Computational Problem Solving section showed a small to medium effect size, suggesting that students' engagement with the content and process positively impacted their problem-solving skills. We found no statistical significance for average scores on Self-Perception (t=0.74, p =0.46).

Discussion and conclusion

Students showed improvement in two out of three different measurements of Computational Thinking after being exposed to the WL curriculum and platform for 8-10 hours of contact time. They improved in Algorithmic Thinking/Computational Problem Solving Practices and Modeling/Simulation practices, as reflected in their learning gain. They also developed complex models for their games and debugged those models as they built their games, which had branching and distinct levels of difficulty, using concepts such as loops and parallelism.

Students did not improve in their Self-Perceptions of their CT abilities, which probed students to think about the level of comfort and ease with which they can engage in computational practices and use digital media for problem-solving. There are multiple possibilities why there was no significant improvement in their self-perceptions of their CT abilities. This might be because of the steep learning curve experienced during the 8-hour curriculum or because it was their first experience as active creators of complex games. This experience might have challenged their self-perception of their technology/digital skills. These findings reflect similar results in active vs. passive STEM learning that suggest novice learners are inaccurate in their judgments of how much they learned; this "feeling of learning" can be negatively correlated with actual learning (Deslauriersa et al., 2019). Future work may include increasing student exposure or contact hours with the curriculum, which could make students more comfortable with the learning curve. Another possibility would be to add specific reflection prompts that encourage students to think about what skills they have developed as they participate in this game design and creation process, thereby deliberately building their awareness of the skills they are developing.

The results of our study show that a curriculum based on game design and play can impact the development of different dimensions of CT. Our findings invite discussion for a more granular analysis of our curriculum activities and their impact. Different measurements and instruments of Computational Thinking, focusing on different aspects, could yield different results, as they capture only a portion of what constitutes thinking computationally. Therefore, assessing CT through a multi-dimensional assessment that looks at CT skill development holistically will lead to a clearer understanding of where our students stand and what remains to be improved upon.

References

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. Educational Technology & Society, 19 (3), 47–57.

Arroyo, I., Closser, A. H., Castro, F., Smith, H., Ottmar, E., & Micciolo, M. (2022). The WearableLearning Platform: A Computational Thinking Tool Supporting Game Design and Active Play. Technology, Knowledge and Learning. https://doi.org/10.1007/s10758-022-09601-1

Castro, F. E. V. (2022). Exploring the Use of Finite-State Machines and Game Creation to Teach Computational Thinking in Middle Schools. Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2, 618. https://doi.org/10.1145/3502717.3532137

Deslauriers, L., McCarty, L. S., Miller, K., Callaghan, K., & Kestin, G. (2019). Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. PNAS, 116(39), 19251–19257. https://www.pnas.org/doi/full/10.1073/pnas.1821936116

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school*, *19*(1), 19-38.

Yasmin B. Kafai & Quinn Burke (2015) Constructionist Gaming: Understanding the Benefits of Making Games for Learning, Educational Psychologist, 50:4, 313-334. https://doi.org/10.1080/00461520.2015.1124022

Kalelioglu, F., & Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. Baltic Journal of Modern Computing, 4, 583-596.

Resnick, M., Maloney, J., Hernández, An., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for Everyone. Communications of the ACM. 52. 60-67.

Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions, 79–98. https://link.springer.com/chapter/10.1007%2F978-981-13-6528-7_6

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying Computational Thinking. Educational Research Review, 22, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003.

van Laar, E., van Deursen, A., van Dijk, J., de Haan, J. (2017) The relation between 21st-century skills and digital skills: A systematic literature review, Computers in Human Behavior, 72, 577-588. https://doi.org/10.1016/j.chb.2017.03.010.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. Association for Computing Machinery, 49(3), 33–35. https://doi.org/10.1145/1118178.1118215

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants #2026722 and #2041785. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.