DEVELOPING A BOARD GAME TO FOSTER COMPUTATIONAL THINKING IN K-12 EDUCATION

by

DAYAE YANG

(Under the Direction of Theodore J. Kopcha)

ABSTRACT

This dissertation follows a manuscript-style format, beginning with an introduction and literature review, followed by three distinct studies and a concluding section. All three studies use "*Lucky Codes*," an educational board game designed to foster computational thinking skills in elementary students, as the intervention. The first article details the development process of the board game with findings from addressing the challenges of balancing learning objectives with enjoyable gameplay and engaging students with higher-level programming concepts. The second article investigates the game's efficacy through a qualitative pilot study with four elementary students. The study demonstrates how the game facilitated student engagement and their understanding of programming concepts. The third article provides a more in-depth analysis of the game through an exploratory case study, using a mixed methods approach to gain a more nuanced understanding of students' game-based learning experiences. The study examines gameplay trends, changes in CT knowledge, and attitude patterns while also looking at gender differences.

Together, these articles offer a comprehensive understanding of how educational board games can be designed, developed, and utilized to support CT education in K-12 classrooms. The collective findings underscore the potential of unplugged games as engaging and effective tools for teaching CT skills, providing practical recommendations for educators and game designers. This dissertation aims to provide deeper insights into the role of board games in CT education, address student needs in game-based learning environments, inform instructional strategies, and guide the development of more effective educational games.

INDEX WORDS: Computational thinking, K-12 education, Educational games, Game-based learning, Computer science education, Coding, Unplugged learning

DEVELOPING A BOARD GAME TO FOSTER COMPUTATIONAL THINKING IN K-12 EDUCATION

by

DAYAE YANG

B.S., Hankuk University of Foreign Studies, The Republic of Korea, 2017

M.Ed., Indiana University, 2019

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2024

DEVELOPING A BOARD GAME TO FOSTER COMPUTATIONAL THINKING IN K-12 EDUCATION

by

DAYAE YANG

DEDICATION

This dissertation is dedicated to those who have provided unwavering support and guidance throughout my academic journey.

To my family—Mom, Dad, Dayoung, and my in-laws—thank you for your endless love, support, and prayers. Your encouragement and support have been the foundation upon which I have built my strength to overcome the challenges of this journey.

To my friends, your companionship and empathy have been a source of much-needed motivation in moments of doubt. Special thanks to Yein for generously sharing your expertise in statistics– you have saved me from countless headaches and, I believe, added a couple of years to my longevity.

To my beloved husband, Shawn, your unwavering love, patience, and support in every way have been my greatest source of strength. You have stood by me through every high and low, and your belief in me has been my greatest encouragement. It would have been impossible to have come this far without you.

This work is a testament to the love, support, and guidance you all have provided. Thank you. I dedicate all glory to God for blessing me with such incredible people in my life and for granting me the wisdom and guidance to persevere through this journey.

ACKNOWLEDGEMENTS

Firstly, I would like to express my deepest gratitude to my advisor, Dr. TJ Kopcha. Your invaluable expertise, encouragement, and patience have been instrumental in guiding me through this process. You have shown me the perfect model of what a good mentor looks like. I am profoundly grateful to have had you as my advisor. I would also like to thank my committee members, Dr. Lloyd Rieber, Dr. Ai-Chu Elisha Ding, and Dr. Kyungbin Kwon. Your invaluable knowledge, guidance, and encouragement have been pivotal throughout this research journey. I deeply appreciate the time and effort you dedicated to reviewing my work and providing thoughtful feedback.

Lastly, I would like to acknowledge Double Helix STEAM School for granting me permission to conduct my research. I am immensely grateful to the teachers for their unwavering support and cooperation. Your flexibility and encouragement significantly eased the data collection process. I also want to thank the students for their enthusiastic participation. Your active participation was crucial to the success of this study.

Thank you all for your tremendous support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

Background

The project began in 2019 with a simple question: "How can a board game be created to foster computational thinking (CT) in a more fun and accessible way for K-12 classrooms?" While the emphasis on CT education was growing, accounts from teachers and literature reviews highlighted numerous challenges that educators frequently encounter when teaching the subject; these challenges will be detailed in the following studies. Based on these ideas, the journey to develop a board game began—a game that would be engaging, effective for learning, and easy to integrate into classrooms, ultimately serving as a practical tool for teachers.

Using available materials in the office—such as pieces of paper, sticky notes, and various items lying around—the initial prototype of the game was created from scratch through multiple brainstorming sessions. These early prototypes underwent several rounds of experimentation and refinement while simultaneously establishing a solid theoretical foundation for the board game. However, rounds of literature review revealed that although there was ample research on digital game-based learning and unplugged CT learning, studies specifically addressing CT board games were relatively scarce. This lack of research provided limited resources and guidance for the development process, making the game development process more challenging than anticipated.

This discovery sparked my passion for contributing to CT board game studies and sharing findings that could provide practical support to educators, researchers, and game developers. Drawing on existing theories and available findings, a fully functional board game, *Lucky Codes,* was developed for the project; the details of the game will be explained in the following studies. This game was used as an intervention in the studies for this dissertation, with the ultimate goal of deepening academic understanding of board games as educational tools.

Introduction

Computational thinking (CT) has emerged as a vital skill in K-12 education, recognized for its role in equipping students with problem-solving abilities and system design skills relevant across various disciplines. Wing (2006) defined CT as using core computer science concepts to solve problems, create systems, and understand human behavior by applying core computer science concepts. Key components of CT include abstraction, sequencing, conditionals, algorithms, loops, parallelism, and debugging (Ezeamuzie & Leung, 2022). Over the past two decades, educators worldwide have increasingly integrated CT into K-12 curricula, reflecting its growing importance in preparing students for a technology-driven world (Lindberg et al., 2019; Nambiar, 2020).

Programming education has been widely adopted as an effective way to introduce CT to young learners, particularly through visual block-based tools like Scratch and Code.org (Lye & Koh, 2014; Resnick et al., 2009). These tools help simplify the programming process, making programming more accessible to students without requiring them to learn complex syntax. Implementing computer-based programming education in K-12 classrooms often faces challenges, including limited teacher expertise in the subject, technological issues with the computers, and the inherently abstract nature of coding concepts, which can be especially challenging for classrooms with younger students (Bell & Vahrenhold, 2018; Boz & Allexsaht-Snider, 2022). This highlights the need to explore alternative methods that engage learners in CT skills beyond traditional computer-based approaches.

One promising approach to teaching CT is the use of board games, which provide an unplugged, hands-on learning experience that makes CT concepts accessible to all students, independent of their access to technology (Bell & Vahrenhold, 2018; Gibson & Bell, 2013). By using tangible pieces, board games help students build concrete understandings of abstract CT concepts, making complex ideas more approachable. This tactile interaction not only enhances individual learning but also fosters peer discussion and collaborative problem-solving, as students can share strategies and learn from one another in a shared, interactive environment (Barsalou & Wiemer-Hastings, 2005; 2018).

Despite these potentials, however, there are still gaps in their design and research that need to be addressed. Many existing CT-focused board games, such as Robot Turtles and CodeMonkey Island, often lack comprehensive programming concepts or rely on simplified, sequential card-stacking mechanics, which creates a significant gap between the gameplay and the actual programming environment (Scirea & Valente, 2020; Wu, 2018). Additionally, current research on educational board games often emphasizes self-reported data, focusing on motivation and user experience rather than examining how specific game mechanics influence the development of CT skills (Apostolellis et al., 2014; Gresse von Wangenheim et al., 2019). These limits leave questions about how game mechanics can effectively support the learning of different CT concepts. Understanding how students engage with CT board games based on varied levels of interest and learning preferences can provide valuable insights for creating more inclusive and effective educational tools.

This dissertation study aims to address these gaps by exploring how specific game mechanics in educational board games affect CT skill development among boys and girls. By examining the learning processes and engagement patterns across genders, this collection of studies aims to contribute to the understanding of educational board games for CT, supporting their effective use in enhancing students' learning experiences and outcomes.

The Terms: Computational Thinking, Computer Science, Programming, and Coding

Since computational thinking has become popular in K-12 schools, the terms computational thinking, computer science, programming, and coding have been frequently used in educational settings such as school curricula, workshops, or after-school programs. However, these terms have often been used interchangeably without a clear distinction of what they mean, and numerous scholars have voiced the need to clarify their meaning (e.g., (Czerkawski & Lyman, 2015; Grover & Pea, 2013; Lu & Fletcher, 2009; Shute et al., 2017). While each term will need its own set of paragraphs to fully explain its origin and how they are being used in different fields, this section will briefly summarize each meaning and their relationship with each other, specifically in the K-12 education context. (See Figure 1.1 for the simplified visual representation of the relationship between the terms.)

Computer science is a field of study that includes learning about programming. *Computer science education*, therefore, consists of not just learning how to program but also learning other skills, such as software engineering or information management. *Computational thinking* is a broader term than computer science because it is a way of thinking that applies to everyday problem-solving, not just to using computers or programming (Shute et al., 2017). *Programming* is a comprehensive process of creating a program that includes coding as well as planning, designing, testing, and deploying. In short, computational thinking includes many parts of computer science, and computer science includes learning about programming; programming is just one way of learning computational skills.

Another frequently used buzzword in recent K-12 education is *coding*. While programming and coding are not the same, practitioners often have used the two words as synonyms (Armoni, 2016; Duncan et al., 2014). By technical definition, *coding* is only a part of programming; it is the act of writing code the computer can translate. Recently, however, the word *coding* has been used more as jargon, especially in K-12 schools, to refer to a "much more playful and non-intimidating description of programming for beginners… to convey the beginning steps of programming, or programming with a tool intended for beginners" (Prottsman, 2017, para. 6).

Using the word *coding* to represent programming could carry the risk of confusing students and educators, leading them to be overconfident about believing that they know all there is to programming when it, in fact, involves much more discipline (Tedre & Denning, 2016). However, there is also an advantage to using the word *coding* for young students in that it carries a hint of mystery (as in having a secret code that could be cracked), which could capture more students' interest (Duncan et al., 2014). Another significant advantage is that it is already used by so many people in K-12 education and has become a common concept that often does not require much explaining (Duncan et al., 2014). Recognizing these benefits and for more accessible communication of the meanings, this dissertation will also use the word coding as a synonym for programming, particularly since the focus of the study is on novice learners within the younger K-12 student population.

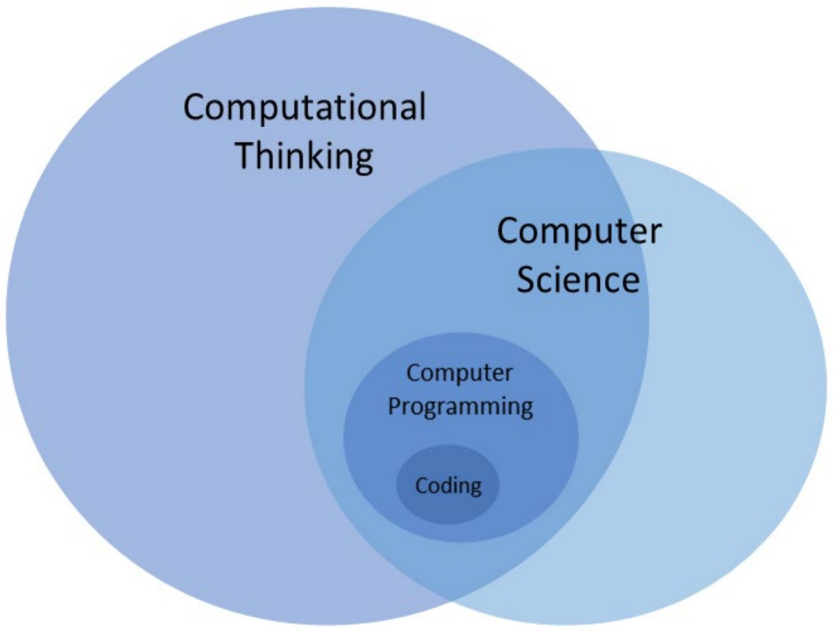


Figure 1.1: Visual representation of the relationships between computational thinking, computer science, computer programming, and coding in K-12 education.

Literature Review

History of Computational Thinking

In the last few decades in the field of K-12 education, computational thinking (CT) has become an essential topic, recognized for its importance in problem-solving and designing systems. In 2006, Jeanette Wing's seminal essay entitled *Computational Thinking* launched a new wave of interest toward the term computational thinking. It spurred the movement to provide computer science courses for all students in K-12 schools. Computational thinking, until that time, was often only viewed as skills required for software development or mathematics and sciences. Wing (2006), however, redefined computational thinking as "a fundamental skill for everyone, not just for computer scientists" (p. 33) that teaches students problem-solving skills that are necessary for everyday life. Wing also believed that computer science education was not an objective but a tool that helped students understand computational thinking; CT was the result of learning computer science.

Following Wing's definition of CT in the K-12 context, scholars have extensively worked to refine and broaden this evolving concept. Early definitions often emphasized technical skills tied closely to computer science, such as solving problems, designing systems, and formulating algorithms (Aho, 2012; Denning, 2009; Wing, 2006). These perspectives typically linked CT with specific programming tasks and the use of computers as problem-solving tools (Barr & Stephenson, 2011; Cuny et al., 2010). As research progressed, the focus shifted toward more generalized problem-solving skills, with less emphasis on technical specifics and computer reliance (Wing, 2006; Yadav et al., 2014). Recent definitions have further expanded the concept to highlight its broad applicability across various domains, emphasizing that computational thinking can be used "with or without the assistance of computers" (Shute et al., 2017, p. 151) and is valuable for "explaining and interpreting the world" (Denning & Tedre, 2021, p. 365). This evolution reflects a growing recognition of computational thinking as a versatile and fundamental skill relevant to diverse educational and real-world contexts beyond its original ties to computer science.

Education in CT has now been widely accepted as a necessity for all students, regardless of age and background. Numerous scholars have called for integrating computational thinking into K-12 education (e.g., Barr & Stephenson, 2011; Grover & Pea, 2013; Lye & Koh, 2014). Educators and administrators worldwide have integrated CT-related subjects into K-12 curricula (Lindberg et al., 2019; Nambiar, 2020). In 2014, England led the way in adding programming as the mandated subject in the K-12 curriculum to nurture students' digital literacy (Wong et al., 2015). Finland soon joined to add computer programming as part of their country's national curriculum, with the belief that the "future will be built by those who know how to code"

(Haaramo, 2014). In 2016, the United States initiated *Computer Science for All* to equip all American students with computer science skills, including coding (Smith, 2016). In subsequent years, more countries around the world, like Australia, Estonia, France, Israel, South Korea, China, Singapore, Taiwan, and Canada, have also started including coding as part of their core national school curricula (Lindberg et al., 2019; Nambiar, 2020; Wu et al., 2020).

As these global efforts illustrate, programming has been widely recognized as an effective way to develop CT skills, including problem-solving, analytical thinking, algorithmic thinking, and critical thinking (Bocconi et al., 2016; Grover et al., 2015; Kafai & Burke, 2014; Lye & Koh, 2014; Resnick et al., 2009; Wong et al., 2015). Moreover, programming also fosters creative thinking, expression, and communication, which are vital for students' overall cognitive and creative development (Strawhacker & Bers, 2019). As the demand for coding skills continues to grow in the job market, learning to code also prepares students for future career opportunities (Nambiar, 2020; World Economic Forum, 2016). Given these advantages, scholars have advocated for introducing coding education at an early age, arguing that early exposure to programming concepts enables students to more readily grasp and apply these skills throughout their academic and professional journeys (Forquesato & Borin, 2018).

Techniques and Strategies

The techniques and strategies for teaching coding have evolved significantly over the years. Traditionally, coding instruction relied heavily on textbooks, where students followed pre-written code examples and made minor modifications, or instructors presented the syntax and features of a specific programming language. However, these methods often left students with a basic familiarity with programming concepts but without the problem-solving skills needed to apply them effectively (Gaspar & Langevin, 2007). Robin et al. (2003) argued that teaching should focus more on the cognitive processes involved in programming rather than solely on conceptual knowledge.

To improve coding instruction, researchers have explored various non-traditional methods. Ali (2005) suggested three strategies for introductory computer science education: scaffolding, concept mapping, and constructivism. *Scaffolding* provides support during the problem-solving process, gradually leading students to independent learning. This can be facilitated through intelligent tutoring systems that offer customized guidance and feedback. *Concept mapping* involves the use of visual symbols or diagrams to represent knowledge, helping students understand complex relationships. *Constructivism* encourages students to actively construct new knowledge by integrating it with what they already know, emphasizing the importance of understanding prior knowledge, refining learning activities, and developing problem-solving skills and assessment methods.

Another widely recognized strategy is live coding, which involves "the process of designing and implementing a [coding project] in front of the class during the lecture period" (Paxton, 2002, p. 52). Studies have supported the benefits of using this approach. For example, Paxton's (2002) and Gaspar and Langevin's (2007) studies found that students preferred live coding more than static lectures, yielding better learning outcomes. Bennedsen and Caspersen (2005) reported that recorded live coding sessions also led to high learning outcomes in online courses. Similarly, Rubin (2013) found that live coding helped students perform better in final projects by modeling good programming habits like iterative testing. They concluded that live coding is as effective, if not better, than traditional teaching, particularly in helping students manage larger assignments.

Specific techniques have also been developed for teaching coding to younger children. A popular way to introduce coding to children is through visual block-based programming tools such as Scratch and Blockly. These tools allow children to learn basic coding skills and then apply them creatively in projects such as stories, games, and quizzes (Bottino et al., 2020; Pinto & Escudeiro, 2014). Kaplancali and Demirkol (2017) identified a structured approach for teaching these fundamental coding concepts, suggesting a two-tiered sequence: first-tier concepts include algorithms, loops, and if-conditionals, followed by second-tier concepts like functions, graphics, variables, and lists.

Another effective approach for teaching young learners is the use of unplugged activities, which teach coding concepts without computers or digital devices. Examples include coding board games like Robot Turtles and Code Master, physical demonstrations of coding actions (e.g., repeating dance moves to illustrate loops), or hands-on activities using physical objects to represent coding concepts (e.g., using envelopes to teach variables). These unplugged activities make coding more accessible to all students, including those without access to technology and help concretize abstract concepts using tangible objects (Bell & Vahrenhold, 2018; Gibson & Bell, 2013). Additionally, unplugged activities foster collaborative learning, as they allow students to work in teams, observe and learn from one another, and actively participate without the distraction of personal devices.

Challenges of Coding Education

Teaching programming is difficult (Grover et al., 2015). Despite the continuing research into effective instructional strategies for coding education, many challenges remain. One major issue is the overall lack of appropriate teaching resources. The available instructional materials for teaching computer programming do not always consider the audience's varying levels of prior knowledge or developmental capabilities (Shanley et al., 2022; Strawhacker & Bers, 2019).

Additionally, there is a shortage of teachers with the necessary background in computer science, leaving many educators unsure about what content to teach or how to integrate coding into other subjects (Boz & Allexsaht-Snider, 2022; Mason & Rich, 2019; Yadav et al., 2014). The scarcity of professional development programs further compounds this problem, as teachers lack opportunities to learn how to incorporate coding into the existing curriculum effectively (Bower & Falkner, 2015; Boz & Allexsaht-Snider, 2022; Mills et al., 2024). As a result, many teachers report feeling unprepared and lack confidence in teaching coding to young students (Bower & Falkner, 2015; Boz & Allexsaht-Snider, 2022; Mason & Rich, 2019). For instance, Sentance and Csizmadia's (2016) survey of 1100 teachers in the UK found that despite spending over 100 hours on self-study and workshops, they still felt inadequately prepared to teach programming, which often led to decreased motivation and negative attitudes toward the subject.

Teachers also face systemic challenges in coding education, including inadequate infrastructure and limited support. Bećirović (2023) and Ntorukiri et al. (2022) identified limited access to technology, insufficient administrative and policy support, and inadequate infrastructure as major barriers to effective technology integration in education, emphasizing the need for professional development and supportive policies. Similarly, Kim et al.'s (2019) investigative report on coding education in schools in South Korea identified several major challenges, including a lack of understanding of coding's importance among teachers and administrators, insufficient access to essential equipment such as tablet PCs or computers, and unreliable network connections, which can lead to overall low-quality instructional support and educational disparities. In Hong Kong, Wong et al. (2015) found that primary teachers struggled with student disinterest, limited lesson planning time, and lack of support from parents and curriculum guidelines, while secondary teachers faced additional challenges related to programming syntax, abstract concepts, and managing diverse skill levels in the classroom. These challenges highlight the need for comprehensive support systems and targeted professional development to help teachers effectively implement coding education.

The Potential of Games in Coding Education

Games offer a promising solution to some of the challenges faced in programming education. They act as flexible instructional materials that support students with diverse abilities by providing built-in scaffolding, such as levels and hints, which allows students to learn at their own pace. Providing such scaffolding effectively supports students with varying abilities within the same learning environment (Pea, 2004; Revelle, 2013). Digital games, especially, often include sequences of levels that gradually increase in difficulty as players become more proficient (Revelle, 2013). They also offer hints or clues to help students who need extra support with solving problems. Beginner students can choose to use these aids and stay at lower levels longer, while advanced students can proceed more quickly to higher levels.

These scaffoldings can also reduce the burden on teachers who may lack sufficient knowledge or strategies, often saving their valuable planning time. Games are dynamic, adaptive, and autonomous, fostering self-regulated learning that allows students to manage their own learning experiences (Neitfeld & Shores, 2011; Nietfeld et al., 2014). Although teachers also need to become familiar with the game in advance to be able to assist students, the built-in support within games often reduces the need for extensive teacher intervention compared to traditional methods.

Another major challenge in programming education is that students often lack the motivation to learn to code. The concepts in coding often require a high level of abstraction, which can be difficult for young learners. Games are well-known for their potential to enhance student motivation and engagement (Mayer, 2014; Kapp, 2012; Plass et al., 2015). Games include features such as incentive systems, competition, instant feedback, and challenges, all of which are known to promote motivation and engagement that are strongly linked to student achievement (Shute et al., 2009). Additionally, games promote motivation through "graceful failure," allowing players to learn through repeated attempts without severe consequences (Plass et al., 2015, p. 261). When playing games, players expect to have failures in the first few attempts. These failures create the necessary challenges to make the tasks more interesting, giving players a higher sense of achievement when they succeed. The low consequences of failures encourage risk-taking, exploration, and trying new strategies (Hoffman & Nadelson, 2010). This makes games particularly effective in coding education, as they stimulate curiosity and challenge, keeping students engaged with abstract programming concepts that might otherwise seem unappealing. As Mladenović et al. (2016) noted, “students are not even aware they learn problem-solving, debugging, and making scenarios; instead, they think they are simply [playing]" (p. 523).

Lastly, games also allow programming to be learned in an unplugged environment, which addresses the challenges related to inadequate technological infrastructure and difficulties in grasping abstract programming concepts. Unplugged games, such as board games or card games, do not require digital devices or internet connection, making them accessible for classrooms without adequate technology. Several unplugged coding games for kids are available in the market, such as Robot Turtles, Code Master, and CodeMonkey Island. There are also numerous online resources, such as Teach Your Kids Code (teachyourkidscode.com), CS Unplugged (https://www.csunplugged.org/), or Hour of Code (https://hourofcode.com/) that provide ideas for using everyday items to create fun unplugged game-based activities for children to learn fundamental programming concepts. Such activities not only make coding education more accessible to classrooms but also allow students to interact with tangible objects, helping to make abstract concepts more concrete (Barsalou & Wiemer-Hastings, 2005). For example, understanding the concepts of variables and values in programming can be particularly challenging for young students due to their abstract nature (Piteira & Costa, 2013). In the *Nursery Rhyme Coding Game* (Siu, 2021), jars and scrap pieces of paper are used to represent variables and values. Each jar is labeled with different variable names like nouns, verbs, and adjectives. Students are to write values on pieces of paper that would go inside the corresponding jars. This physical representation helps young learners understand abstract ideas more easily, demonstrating how tangible objects can enhance the comprehension of programming concepts.

Purpose Statement

The purpose of this dissertation is to explore the design, implementation, and impact of a board game developed for K-12 classrooms to enhance computational thinking (CT) skills. By combining three distinct but related studies, this work aims to contribute to understanding how unplugged educational games can be effectively integrated into classroom settings to support CT learning, focusing on student engagement and interest, with consideration of gender differences.

Dissertation Structure

This dissertation employs a multiple-article structure that follows the University of Georgia Graduate School guidelines (2024) for a manuscript-style dissertation. Based on the guidelines, this style of dissertation includes an introduction and literature review chapter, three chapters formatted as articles intended for publication in peer-reviewed scholarly journals, and a concluding chapter that brings together the significant findings and implications of the overall study. The primary reason for the multiple-article structure is that it allows for a comprehensive exploration of the different facets of the research topic, each addressed through distinct but interconnected studies. This format provides the flexibility to delve deeply into the design, implementation, and impact of the educational board game while also enabling the inclusion of detailed analyses on specific aspects, such as practical application in classroom settings and the consideration of gender differences in computer science education.

Table 1.1 illustrates the connection between the three studies discussed in Chapters 2 to 4 and how each article contributed to the development of the next article. Article 1 (Chapter 2), entitled *Designing a Board Game for Beginning Block-Based Programmers,* presents the design story of the board game, detailing the concepts, theories, and iterative processes behind the creation of earlier prototypes, along with initial playtesting to gather feedback. This foundational study established the educational rationale and design principles that guided the refinement of the game and shaped the research questions for the second article.

Article 2 (Chapter 3), entitled *Using a Board Game for Computer Programming Education*, presents the final version of the game's development and centers on its initial efficacy evaluation. The study includes a pilot study with four students, video recordings, transcriptions, and analyses of student engagement and their understanding of programming concepts. Building on the design principles established in Article 1, this study offered preliminary evidence of the game's impact on learning, informing further evaluation and refinement for the next article.

Article 3 (Chapter 4) expands on these insights through a comprehensive case study with 17 students across three days of gameplay sessions. The study involved multiple data sources, including pre- and posttests, surveys, observations, and interviews. The focus was on identifying gameplay trends, changes in knowledge, and gender differences. Drawing on insights from the game's initial design and testing stages detailed in the previous articles, this study aimed to deepen the understanding of students' interaction and experiences with the board game with additional consideration of gender differences.

Table 1.1: Overview of three studies.

Article		Study Overview	Role
1	Designing a Board Game for Beginning Block-Based Programmers (Published; see Yang & Kopcha, 2022)	A design case of a board game for block-based programming; methods include conceptualization, initial playtesting, and feedback collection.	This article laid the foundation for the board game's design and development, detailing the iterative process of balancing gameplay, learning, and player engagement. It focused on refining game mechanics to effectively teach advanced programming concepts like conditionals and loops. The design insights from the findings shaped the research questions of the second article, which examined how these mechanics support and enhance children's programming skills during gameplay.
2	Using A Board Game for Computer Programming Education: A Qualitative Exploratory Study (Under review)	Evaluation of initial efficacy of the game and its learning effects; methods include a pilot study with four students, video recording and transcription.	This article evaluates how the design principles from Article 1 impacted students' CT skill development through gameplay. Key findings reveal that the game's mechanics, particularly incentivizing the use of conditionals and loops, increased student engagement with these advanced programming concepts. These insights laid the groundwork for Article 3, which further examines the impact of extended gameplay and gender differences.

3	How Do Boys and Girls Learn Computational Thinking Through a Board Game? A Case Study of	A case study of *Lucky Codes* with 17 students across three days; methods include multiple data sources, pre- and posttests, surveys, observations, and interviews. The analysis focuses on gameplay trends, knowledge changes, and gender differences.	This article expands on Article 2's findings by conducting a more detailed examination of how extended gameplay affects students' engagement and learning, with a particular focus on gender differences. The main finding indicates that boys and girls showed distinct patterns in their engagement and CT skill development, highlighting how specific game mechanics influence learning and gameplay experience differently across genders.

References

Aho, A. V. (2012). Computation and computational thinking. *The computer journal, 55*(7), 832-835. https://doi.org/doi: 10.1093/comjnl/bxs074

Ali, S. (2005). Effective Teaching Pedagogies for Undergraduate Computer Science. *Mathematics & Computer Education, 39*(3), 243-257. Retrieved from https://www.proquest.com/scholarly-journals/effective-teaching-pedagogies-undergraduate/docview/235807504/se-2

Althouse, A. D. (2016). Adjust for Multiple Comparisons? It's Not That Simple. *The Annals of Thoracic Surgery, 101*(5), 1644-1645.

Apostolellis, P., Stewart, M., Frisina, C., & Kafura, D. (2014). RaBit EscAPE: a board game for computational thinking. *Proceedings of the 2014 Conference on Interaction Design and Children.* Aarhus, Denmark. https://doi.org/10.1145/2593968.2610489

Armoni, M. (2016). COMPUTING IN SCHOOLS Computer science, computational thinking, programming, coding: the anomalies of transitivity in K-12 computer science education. *ACM Inroads, 7*(4), 24-27. https://doi.org/10.1145/3011071

Ballard, E. D., & Haroldson, R. (2021). Analysis of computational thinking in Children's literature for K-6 students: Literature as a non-programming unplugged resource. *Journal of Educational Computing Research, 59*(8). https://doi.org/10.1177/07356331211004048

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Acm Inroads, 2*(1), 48-54. https://doi.org/10.1145/1929887.1929905

Barsalou, L. W., & Wiemer-Hastings, K. (2005). Situating abstract concepts. In D. Pecher, & R. A. Zwaan (Eds.), *Grounding cognition: The role of perception and action in memory,* *language, and thinking* (pp. 129-163). Cambridge University Press. https://doi.org/10.1017/CBO9780511499968.007

Bećirović, S. (2023). Challenges and Barriers for Effective Integration of Technologies into Teaching and Learning. In *Digital Pedagogy.* Singapore: Spinger. https://doi.org/10.1007/978-981-99-0444-0_10

Bell, T., & Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work? In H. Böckenhauer, D. Komm, & W. Unger (Eds.), *Adventures Between Lower Bounds and Higher Altitudes* (pp. 497–521). Springer, Cham. https://doi.org/10.1007/978-3-319-98355-4_29

Bennedsen, J., & Caspersen, M. E. (2005). Revealing the programming process. *Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE '05)* (pp. 186-190). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1047344.104741

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education - Implications for policy and practice.* Joint Research Center (European Commission). Retrieved from https://data.europa.eu/doi/10.2791/792158

Bottino, R., Chioccariello, A., & Freina, L. (2020). Computational Thinking in Primary School Through Block-Based Programming. In D. Burgos (Ed.), *Radical Solutions and eLearning. Lecture Notes in Educational Technology.* Singapore: Springer. https://doi.org/10.1007/978-981-15-4952-6_10

Bower, M., & Falkner, K. (2015). Computational Thinking, the Notional Machine, Pre-service Teachers, and Research Opportunities. *Proceedings of the 17th Australasian Computing*

Education Conference (ACE 2015). 27, p. 30. Sydney, Australia: Australian Computer Society.

Boz, T., & Allexsaht-Snider, M. (2022). How do elementary school teachers learn coding and robotics? A case study of mediations and conflicts. *Education and Information Technologies, 27*(3), 3935-3963. https://doi.org/10.1007/s10639-021-10736-4

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress, referenced in http://www. cs. cmu. edu/~ CompThink/resources/TheLinkWing. pdf.*

Czerkawski, B., & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. *TechTrends, 59*, 57-65. https://doi.org/10.1007/s11528-015-0840-3

Denning, P. J. (2009). The profession of IT Beyond computational thinking. *Communications of the ACM, 52*(6), 28-30. https://doi.org/http://doi.acm.org/10.1145/1516046.1516054

Denning, P. J., & Tedre, M. (2021). Computational Thinking: A Disciplinary Perspective. *Informatics in Education, 20*(3), 361-390. Retrieved from https://www.infedu.vu.lt/journal/INFEDU/article/701/file/pdf

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn coding? *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14).* New York, NY: Association for Computing Machinery. https://doi.org/10.1145/2670757.2670774

Ezeamuzie, N. O., & Leung, J. S. (2022). Computational thinking through an empirical lens: A systematic review of literature. *Journal of Educational Computing Research, 60(2)*, 481-511. https://doi.org/07356331211033158

Forquesato, L. E., & Borin, J. F. (2018). Kids Block Coding Game: A game to introduce programming to kids. *Proceedings of the XXVI Workshop on Computer Education.* Porto Alegre, Brazil: SBC. https://doi.org/10.5753/wei.2018.3502

Gaspar, A., & Langevin, S. (2007). Restoring" coding with intention" in introductory programming courses. *Proceedings of the 8th ACM SIGITE conference on Information technology education (SIGITE '07)* (pp. 91-98). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1324302.1324323

Gibson, B., & Bell, T. (2013). Evaluation of games for teaching computer science. *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, (pp. 51–60). Aarhus, Denmark. https://doi.org/10.1145/2532748.2532751

Gresse von Wangenheim, C., Araújo e Silva de Medeiros, G., Missfeldt Filho, R., Petri, G., da Cruz Pinheiro, F., Ferreira, M. N., & Hauck, J. C. (2019). SplashCode - A Board Game for Learning an Understanding of Algorithms in Middle School. *Informatics in Education, 18*(2), 259-280. https://doi.org/10.15388/infedu.2019.12

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer science education, 25*(2), 199-237. https://doi.org/10.1080/08993408.2015.1033142

Haaramo, E. (2014, July 11). *Future will be built by those who know how to code*. Retrieved from SITRA: https://www.sitra.fi/en/articles/future-will-be-built-those-who-know-how-code/

Hoffman, B., & Nadelson, L. (2010). Motivational engagement and video gaming: A mixed methods study. *Educational Technology Research and Development, 58*(3), 245-270.

https://doi.org/10.1007/s11423-009-9134-9

Hong, J. C., Hwang, M. Y., Wu, N. C., Huang, Y. L., Lin, P. H., & Chen, Y. L. (2016). Integrating a moral reasoning game in a blended learning setting: effects on students' interest and performance. *Interactive Learning Environments, 24*(3), 572-589. https://doi.org/10.1080/10494820.2014.908926

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming.* MIT Press.

Kaplancali, U. T., & Demirkol, Z. (2017). Teaching coding to children: A methodology for kids 5+. *International Journal of Elementary Education, 6*(4), 32-37. https://doi.org/10.11648/j.ijeedu.20170604.11

Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education.* John Wiley & Sons.

Kim, Y., Yoo, J., & Kim, N. (2019). *The Current Status and Improvement Tasks of Software Education in Elementary and Secondary Schools [초·중등 소프트웨어교육 운영실태와 개선과제]*. National Assembly Research Service. Retrieved from https://www.nars.go.kr/report/view.do?categoryId=&cmsCode=CM0043&searchType=TITLE&searchKeyword=&brdSeq=26770

Krippendorff, K. (2013). *Content Analysis: An Introduction to Its Methodology.* Thousand Oaks, CA: Sage Publications.

Lindberg, R. S., Laine, T. H., & Haaranen, L. (2019). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games. *British journal of educational technology, 50(4)*, 1979-1995. https://doi.org/10.1111/bjet.12685

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. *Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09)* (pp. 260-264). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1508865.1508959

Lye, S. Y., & Koh, J. H. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. https://doi.org/j.chb.2014.09.012

Mason, S. L., & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education, 19*(4), 790-824. Retrieved from https://www.learntechlib.org/primary/p/184723/.

Mayer, R. E. (2014). *Computer games for learning: An evidence-based approach.* MIT Press.

Mills, K. A., Cope, J., Scholes, L., & Rowe, L. (2024). Coding and Computational Thinking Across the Curriculum: A Review of Educational Outcomes. *Review of Educational Research*. https://doi.org/10.3102/00346543241241327

Mladenovic, S., Krpan, D., & Mladenović, M. (2016). Using Games to Help Novices Embrace Programming: From Elementary to Higher Education. *International Journal of Engineering Education, 32*(1), 521-531.

Monga, M., Lodi, M., Malchiodi, D., Morpurgo, A., & Spieler, B. (2018). Learning to program in a constructionist way. *Proceedings of Constructionism 2018.* Vilnius, Lithuania. Retrieved from https://inria.hal.science/hal-01913065

Nambiar, R. (2020). Coding as an Essential Skill in the Twenty-First Century. In B. Panth, & R. Maclean (Eds.), *Anticipating and Preparing for Emerging Skills and Jobs* (pp. 237-243).

Springer, Singapore. https://doi.org/10.1007/978-981-15-7018-6_29

Neitfeld, J., & Shores, L. R. (2011). Self-regulation within game-based learning environments. In L. Annetta, & S. C. Bronack (Eds.), *Serious Educational Game Assessment: Practical Methods* (pp. 19-42). Leiden, The Netherlands: Brill. Retrieved from https://brill.com/view/book/edcoll/9789460913297/BP000003.xml

Nietfeld, J. L., Shores, L. R., & Hoffmann, K. F. (2014). Self-Regulation and Gender Within a Game-Based Learning Environment. *Journal of Educational Psychology, 106*(4), 961. Retrieved from http://www.johnnietfeld.com/uploads/2/2/6/0/22606800/nietfeld_shores_hoffmann_2014_jep.pdf

Ntorukiri, T., Kirugua, J., & Kirimi, F. (2022). Policy and infrastructure challenges influencing ICT implementation in universities: a literature review. *Discover Education, 1*(19). https://doi.org/ https://doi.org/10.1007/s44217-022-00019-6

Paxton, J. (2002). Live programming as a lecture technique. *Journal of Computing Sciences in Colleges, 18*(2), 51-56. https://doi.org/10.5555/771322.771332

Pea, R. D. (2004). The social and technological dimensions of scaffolding and related theoretical concepts for learning, education, and human activity. *Journal of Learning Sciences, 13*(3), 423-451. https://doi.org/10.1207/s15327809jls1303_6

Pinto, A., & Escudeiro, P. (2014). The use of Scratch for the development of 21st century learning skills in ICT. *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1-4). Barcelona, Spain: IEEE. https://doi.org/10.1109/CISTI.2014.6877061

Piteira, M., & Costa, C. (2013). Learning computer programming: study of difficulties in learning programming. *Proceedings of the 2013 International Conference on Information Systems and Design of Communication (ISDOC '13)* (pp. 75-80). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2503859.2503871

Plass, J. L., Homer, B. D., & Kinzer, C. K. (2015). Foundations of game-based learning. *Educational Psychologist, 50*(4), 258-283. https://doi.org/10.1080/00461520.2015.1122533

Prottsman, K. (2017, Dec 6). Coding vs. Programming -- Battle of the Terms! HuffPost. Retrieved Oct 4, 2024, from https://www.huffpost.com/entry/coding-vs-programming-bat_b_7042816

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Revelle, G. (2013). Applying developmental theory and research to the creation of educational games. *New Directions for Child and Adolescent Development, 2013*, 31-40. https://doi.org/10.1002/cad.20029

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education, 13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Rothman, K. J. (1990). No Adjustments Are Needed for Multiple Comparisons. *Epidemiology, 1*(1), 43-46.

Rubin, M. J. (2013). The effectiveness of live-coding to teach introductory programming. *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)* (pp. 651-656). New York, NY, USA: Association for Computing

Machinery. https://doi.org/10.1145/2445196.2445388

Scirea, M., & Valente, A. (2020). Boardgames and computational thinking: How to identify games with potential to support CT in the classroom. *Proceedings of the 15th International Conference on the Foundations of Digital Games.* Bugibba, Malta. https://doi.org/10.1145/3402942.3409616

Sentance, S., & Csizmadia, A. (2016). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies, 22*(2), 469-495. https://doi.org/10.1007/s10639-016-9482-0

Shanley, N., Martin, F., Hite, N., Perez-Quinones, M., Ahlgrim-Delzell, L., Pugalee, D., & Hart, E. (2022). Teaching Programming Online: Design, Facilitation and Assessment Strategies and Recommendations for High School Teachers. *TechTrends, 66*, 483-494. https://doi.org/10.1007/s11528-022-00724-x

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

Shute, V. J., Ventura, M., Bauer, M., & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning: Flow and grow. In U. Ritterfeld, M. J. Cody, & P. Vorderer (Eds.), *Serious Games: Mechanisms and Effects* (pp. 295-321). Routledge.

Siu, K. (2021, Jan 4). *All About Variables: A Silly Nursery Rhyme Coding Game*. Retrieved 11 12, 2021, from Teach Your Kids Code: https://teachyourkidscode.com/variables-coding/

Smith, M. (2016, Jan 30). *Computer Science For All*. Retrieved from The White House President Barack Obama: https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Strawhacker, A., & Bers, M. (2019). What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development, 67*, 541-575. https://doi.org/10.1007/s11423-018-9622-x

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16)* (pp. 120-129). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2999541.2999542

University of Georgia Graduate School. (2024, May). Theses and dissertations: Student guide to preparation and processing. Athens, Georgia, USA.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49(3)*, 33-35. https://doi.org/1118178.1118215

Wong, G. K., Cheung, H. Y., Ching, E. C., & Huen, J. M. (2015). School perceptions of coding education in K-12: A large scale quantitative study to inform innovative practices. *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 5-10). Zhuhai, China: IEEE. https://doi.org/10.1109/TALE.2015.7386007/

World Economic Forum. (2016). *The future of jobs: employment, skills and workforce strategy for the Fourth Industrial Revolution*. Retrieved from VOCEDplus: https://www.voced.edu.au/content/ngv%3A71706

Wu, L., Looi, C. K., Multisilta, J., How, M. L., Choi, H., Hsu, T. C., & Tuomi, P. (2020). Teacher's perceptions and readiness to teach coding skills: a comparative study between Finland, Mainland China, Singapore, Taiwan, and South Korea. *The Asia-Pacific Education Researcher, 29*(1), 21-34. https://doi.org/10.1007/s40299-019-00485-x

Wu, S. (2018). The Development and Challenges of Computational Thinking Board Games. *2018 1st International Cognitive Cities Conference (ICS3)*, (pp. 129-131). Okinawa, Japan. https://doi.org/10.1109/IC3.2018.00-45

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), 1-16. https://doi.org/10.1145/2576872

Yang, D., & Kopcha, T. J. (2022). Designing a Board Game for Beginning Block-Based Programmers. *International Journal of Designs for Learning, 13*(1), 35-45. https://doi.org/10.14434/ijdl.v13i1.32211

Zallio, M. (2021). Democratizing Information Visualization. A Study to Map the Value of Graphic Design. In M. Soares, E. Rosenzweig, & A. Marcus (Ed.), *Design, User Experience, and Usability: UX Research and Design (HCII 2021)* (pp. 495-508). Springer, Cham. https://doi.org/10.1007/978-3-030-78221-4_33

CHAPTER 2

DESIGNING A BOARD GAME FOR BEGINNING BLOCK-BASED PROGRAMMERS[1]

[1] Yang, D. and Kopcha, T. 2022. *International Journal of Designs for Learning.* 13(1): 35-45.
Reprinted here with permission of the publisher. Copyright © 2024 by the International Journal of Designs for Learning, a publication of the Association of Educational Communications and Technology (AECT) and Indiana University. Permission to make digital or hard copies of portions of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page in print or the first screen in digital media. Except as otherwise noted, the content published by IJDL is licensed under **CC BY-NC-ND 4.0**.

Abstract

Computer programming has become an essential part of K-12 education, promoted as a way for students to engage in computational thinking that helps develop students' ability to analyze and solve problems and prepare them for future careers. Tabletop board games are seen as an effective means to help students learn computer programming. Several board games have been developed for teaching computer science to novice students. Still, many are dominated by simple pathfinding movements lacking comprehensive use of various computer programming concepts or have a considerable gap between the game dynamics and the actual coding that takes place on the computer. This paper presents a design case in which we used Kalmpourtzis' (2018) elements of educational game design (game elements, learning, and players) to develop a board game that engages players who are learning block-based computer programming. We present the four major prototypes and the challenges for each step. Then, we highlight three main areas in which our design process offers implications for the design of educational board games.

Introduction

Computer programming has become an important part of K-12 education, promoted as a way for students to engage in computational thinking that helps develop students' ability to analyze and solve problems effectively and efficiently (Shute et al., 2017; Wing, 2006) and to prepare them for future careers (Gresse Von Wangenheim et al., 2019). Research suggests that learning computer programming skills early at a young age can improve both their skill development and interest in computer science in the future (Bers et al., 2014; Sharma et al., 2019; Tsan et al., 2018). However, learning computer science and programming can be a daunting process for novice learners; it requires multiple higher-level thinking skills and an understanding of abstract concepts (Mcgettrick et al., 2005; Piteira & Costa, 2012; Robins et al., 2003). Students must learn the language of the computer, conceptualizing how different commands and codes will manifest when a program is running.

Adding to the challenge of introducing computer science to younger students is an overall lack of materials and teacher training. The available instructional materials for teaching computer programming do not always take into account the developmental needs and capabilities of the audience (Walton-Hadlock, 2008), resulting in resources and activities that are too difficult for students to accomplish without intervention. In addition, teachers often lack the background knowledge and skills needed to teach children to program a computer (Bell et al., 2009). This is due, in part, to a lack of teacher-preparation programs that help novice teachers learn and teach computer programming (Yadav et al., 2016). Although several block-based programming languages for kids have been developed (e.g., Scratch or Blockly), many teachers continue to express a lack of confidence and concerns with having to teach such an abstract language on the computer to young students (Ericson et al., 2016; Thompson et al., 2013).

Tabletop board games are seen as an effective way to help young children learn computer programming. The use of games in learning, in general, has been shown to enhance students' interest, engagement, and motivation (Apostolelliset al., 2014; Barab et al., 2010; Barata et al., 2013; Nah et al., 2014). Games also provide learners with an opportunity to focus on a small set of essential skills that can be mastered in a friendly, engaging manner (Barab et al., 2010). In the context of computer science, board games not only make learning accessible for anyone with or without the digital infrastructure (Bell & Vahrenhold, 2018; Gibson & Bell, 2013) but also are especially effective in helping younger players develop concrete representations of abstract concepts (Hinebaugh, 2009). This is particularly important when dealing with complex concepts such as algorithms, conditionals, and loops that tend to be overwhelming not only for young children to understand but also for adults to teach children (Fotaris et al., 2016, Olsson et al., 2015).

Several board games have been developed for teaching CT and computer science to children - commercial games such as Robot Turtles and Codemaster, and a few that were developed for research purposes (e.g., Apostolellis et al., 2014; Gresse von Wangenheim et al., 2019; Tsarava et al., 2018). These games support computer programming in that they require players to use a combination of codes (e.g., move forward, left, right) or patterns to manipulate objects on the game board and achieve game goals. However, current challenges with such games are that they are generally often dominated by path movement as the primary game dynamic – that is, through simple sequencing of directions rather than a comprehensive use of various computer programming concepts. This dynamic presents a considerable gap between the game and the actual coding that takes place on the computer (Wu, 2018). Despite the popular demand for such games, little attention has been spent in the literature on the process of designing and developing educational games (Gresse von Wangenheim et al., 2019), still leaving many questions about how a board game that teaches computer programming skills should be designed.

This paper presents a design case in which we used Kalmpourtzis' (2018) elements of educational game design (game elements, learning, and players) to develop a board game to teach young children basic computer programming concepts and skills. The impetus for the project came from a practical issue repeatedly noted by the authors. The online programming environments, Scratch and Scratch Jr., are promoted as essential tools for teaching computer programming skills. However, in our work with local schools, we noticed that both teachers and students often needed an introduction to block-based programming before they could be successful with the available online tools. This gave us the idea for the board game – our thinking was that we could address this practical issue by creating a safe environment in which children and their teachers could learn to use a limited set of block-based codes before moving to an online programming environment. We hoped that the game would help children transition to more sophisticated programming in the future.

With that context in mind, the main goal of the board game was to introduce beginning programmers to the block-based language and basic programming concepts used in Scratch. We had two target audiences in mind for the game. The first was children from grades three to five; the game was designed to help players practice the concepts found in the Computer Science Teachers Association (2017) standards for grades three to five, such as sequencing movements, looping or repeating subroutines, and if-then conditionals. We also thought that the game and game elements would be useful for adults who were just learning about block-based programming. Thus, our second audience was adults interested in learning block-based programming basics so they could support young children in learning CS skills (e.g., novice teachers). Table 2.1 describes how various programming concepts were incorporated into the game.

Table 2.1: Basic programming concepts that were incorporated in the game.

CS CONCEPTS	IMPLEMENTATION IN THE GAME
Sequences	Players will create and follow codes in a sequence to make their character move in favor of the player's strategy (e.g., deciding to turn right first, then move three steps forward to avoid an obstacle and move towards a reward).
Algorithms	With limited access to code tiles, players will consider and compare different ways to solve a task and decide which would be the best solution for them.
Loops	With a limited number of code tiles in their hands, players will use

	"repeat __ times" to repeat a sequence of codes multiple times rather than listing the same commands repeatedly.
Conditionals	Players will use "if __ then" statements to move across obstacles or collect more rewards when the condition is met.

The design process involved four iterations of playtesting. The first three playtests were conducted with graduate students in the field of Learning, Design, and Technology; for the fourth playtest, both graduate students and a 5th-grade student at a local school played the game. We intentionally began our playtesting with graduate students because we were concerned that children might not tolerate the flaws in the gameplay and learning dynamics inherent in our initial design. The graduate students had little to no prior experience with Scratch programming. Our goal was to work first with adults with an underlying background in learning design to refine the game and learning dynamics. This would, in turn, help us make better use of our time when prototyping with young children.

In the first playtest, the players identified critical issues with the initial game core, gameplay, and learner dynamics. The second playtest occurred after making a major change in the game mechanics. Once the game mechanics were working, the third playtest focused on improving the aesthetics of the game and the learning dynamics. The fourth playtest, which took place most recently, focused on making the game enjoyable while also engaging players in higher-level computer programming concepts.

In the sections that follow, we present our initial design and detail the development process across the four prototypes, highlighting the important challenges that were faced during an educational game design process. We then suggest implications for future designers.

The Game Core

The "game core" is a term used by George Kalmpourtzis in his book *Educational Game*

Design Fundamentals (2018). The game core represents the essential intent and purpose for the game being designed. According to Kalmpourtzis, establishing the game core before the creation process is fundamental:

By figuring out their game core, game designers establish a concrete starting point that will help them avoid future issues like technical restrictions, communication problems, finding materials that will affect deadlines, budget, member motivation, and the final impact of the designed game (p. 117).

We chose Kalmpourtzis' (2018) perspectives to guide our design process. Kalmpourtzis suggested that the game core be established by focusing on three essential aspects of educational game design: (1) the players, (2) the game elements and mechanics, and (3) the learning that will take place. He stated that these three aspects are important because a successful educational game designer must consider who the players will be as well as their preferences and learning goal. These factors would ultimately influence what game elements will be included and how the game mechanics will address the desired learning objectives. Based on this idea, our goal as designers was to balance these essential aspects so that the game is fun while ensuring that the players learn from the game.

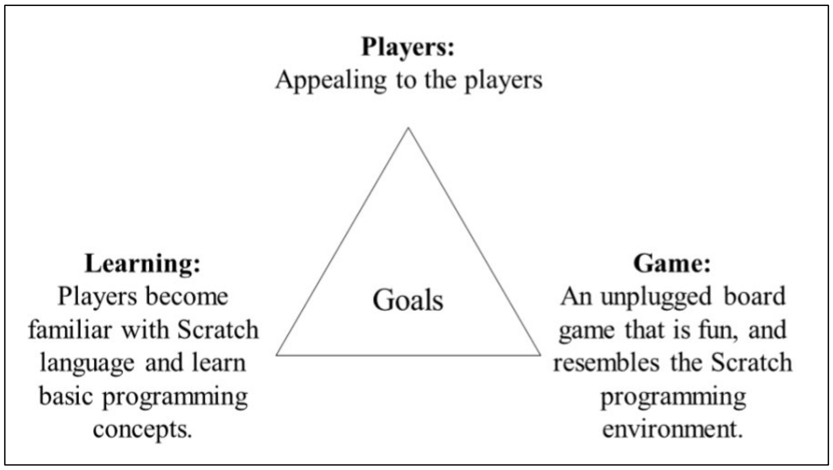

Players:
Appealing to the players
Learning:
Players become familiar with Scratch language and learn basic programming concepts.
Goals
Game:
An unplugged board game that is fun, and resembles the Scratch programming environment.

Figure 2.1: The goals for our board game in the context of Kalmpourtzis' (2018) Game Core.

With Kalmpourtzis' perspectives in mind, our game design process began by identifying the general objectives within each of the three core aspects (see Figure 2.1). First, we wanted to create a board game that would be appealing to both adults and children (*players*) who were learning basic and advanced programming concepts, including conditionals and loops *(learning)*, in a way that was fun and resembled the Scratch programming environment *(game)*. Note that the three elements are interrelated, so it is not always easy to make a clear distinction between each perspective during game design (Kalmpourtzis, 2018). It is important that the designers keep the framework in mind during the initial stages of educational game design; they serve as guidelines that help designers attend to each of the critical aspects of game design, considering how each major design decision affected each aspect, and consciously finding the optimal balance between game, player, and the content to be learned.

In the following sections, we present a design case associated with creating an educational board game that achieved our goal of supporting beginning programmers in learning computer science. The case focuses on how we, as designers, engaged in the process of prototyping that attempted to balance our goals for the game, players, and learning. Below, we present each of the four main prototyping sessions, including the problems that arose from each prototype and our design decisions that helped resolve those problems in each subsequent version. (See Appendix for the summary of challenges in each prototype.)

Prototype 1: Brainstorming

The overall focus of the first prototype was on developing and testing game mechanics. Playtesting our initial design revealed issues with the complexity of the game mechanics and a need to make gameplay more engaging for players. These issues are described in detail below in terms of the focal point of our design and the results of the prototyping session.

Design Focus

Development of the first prototype began by brainstorming the gameplay mechanics. The prototype was constructed simply using markers, crayons, sticky notes, and readily available objects. Because our target audience was players who were new to programming, the concepts used in the game were intended to be intuitive and easy to understand without any prior knowledge.

The earliest idea for gameplay was to move robots on a grid (see Figure 2.2). Although we wanted the game to incorporate more advanced programming concepts, we began with movement as it allows novice players to mimic how characters are programmed to move in digital settings and sequence a computer program in a way that makes intuitive sense. Since one of our objectives was to help such players become familiar with the Scratch programming environment, we used words and functions typically associated with Scratch, such as moving a specific quantity and turning to face the desired direction.

The game board was constructed on a grid so that players could assemble code cards (i.e., the sticky notes in Figure 2.2) to program their robots (round toys) to move and avoid obstacles (crayons) while collecting rewards (blocks) and points (colored spaces). We also included placeholders for constructing blocks of commands. These include spaces for: taking direct action, constructing variables, and constructing repeatable functions. Our thinking was that these would support the *learning* aspect of the game core. The spaces on the gameboard invited players to apply basic concepts (i.e., movement) while offering space for more advanced programming concepts (i.e., loops, conditionals).

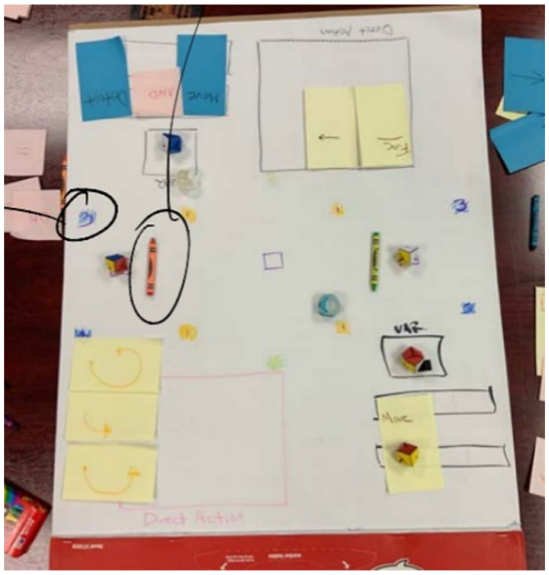


Figure 2.2: The first prototype of our game used readily found objects (i.e., crayons, sticky notes) to draw game board layouts and represent game objects.

Results

Two major issues were found. The first dealt with the balance between the *players* and the *learning*: it was difficult for the players to understand what each placeholder meant. We anticipated that this difficulty would create a high entry barrier for beginning players to fully understand the game and find the game fun to play. The second issue dealt with the *game:* there was not enough space for the robots to move around *and* the players to assemble block codes on a single game board. Both of these issues became the focus of our improvements in our second prototype.

Prototype 2: From One to Two Game Boards

Our work on the second prototype (Figure 2.3) began with a focus on the *game* and the *players.* Our goal was to improve the game board design so that game mechanics were more intuitive for players and that there was more room for gameplay. A major design decision in the second prototype was to distribute the game mechanics across two separate game boards - one for computer programming and one for the movement of the robot. Playtesting revealed that the gameplay significantly improved for the players but did not address the learning of computer programming in a substantial way. Detailed issues are described below in terms of the focal point of our design and the results of the prototyping session.

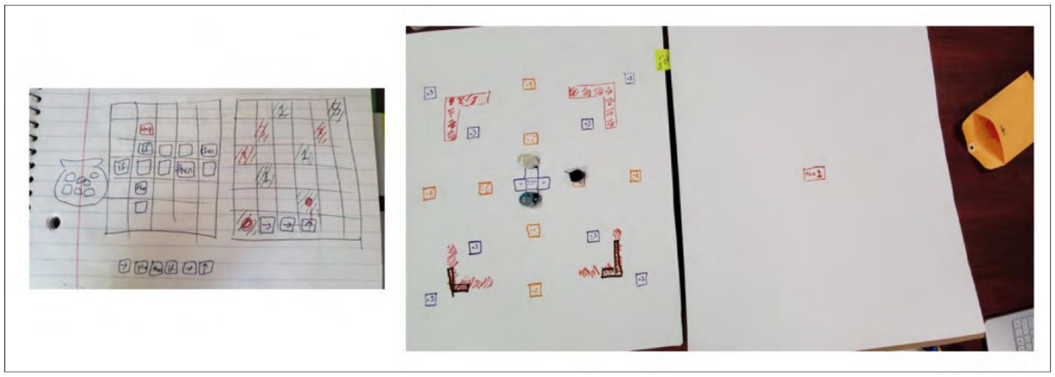


Figure 2.3: Design sketches of our ideas for having two game boards at once, where one board was for coding and the other for the robot to move around.

Design Focus

Our first prototyping session revealed that players felt constrained by the physical aspects of the game board -- there was not enough room on the board for players to construct blocks of code *and* simulate the movement of the robot. To address this issue, we ultimately separated the construction of code from the actions of the robot. As shown in Figure 2.3, gameplay would now take place across two game boards: (1) an "action board" where the players would move the robot, and (2) a "coding board" where players would construct blocks of code. We were inspired by *Scrabble* and *Rummikub* in that we used tiles on a grid that could be modified by all participating players. Each tile would become part of a code block, and the players could combine the tiles to build statements to make their characters move toward a goal.

We anticipated that our design decision about the *game* (i.e., use two game boards) would address the two other components of the game core: the *player's* experience and the *learning* components of the game. Having two game boards would help reduce the confusion created in the first prototype from having multiple areas on a single board for different types of programming skills (e.g., variables, functions). In turn, this would make it easier for players to develop effective blocks of code from their first move in the game (i.e., *learning*). It also added more space for characters to move the robot around while more closely resembling the Scratch programming environment, which provides users a separate space for building codes.

Results

The second prototype was playtested with three graduate students who had no or little experience with Scratch. The participants agreed that using tiles to make codes was a unique and fun experience. However, the players' use of tiles was limited to basic moving and turning movements. Rather than engage in higher-level programming concepts such as conditionals, they were primarily focused on collecting immediate rewards using simple movements. The players also were unclear of the game rules during the play, which significantly slowed down the game. The challenge of finding the balance between the *game elements* (finding a good game mechanic) that can include the necessary *learning objectives* (advanced programming concepts) while making the game still appealing for the targeted *players* (fun and easy to understand) remained.

However, we were not sure at this point whether such a challenge was due to the game design itself or its loose structure and lack of an appealing story or visual design. Such missing parts seemed to limit the motivation of players to play the game long enough to move on from using simple movements to making more complex strategies using advanced statements. Hence, it was necessary to add in the aesthetic elements for our next step to find out if the game did have the potential to prompt players to use advanced statements or if there was a fundamental flaw with the overall game design.

Prototype 3: Making The Game Appealing

The main focus of the third prototype, shown in Figure 2.4, was twofold. First, we sought to make the game more playable and appealing through improved structure and aesthetics. In addition, we sought to improve players' use of higher-level computer programming concepts; this was a recurring issue from previous prototypes. The playtesting of this prototype eventually revealed important information about two aspects of the game core: the *players* and the *learning.* Specifically, the players were much more engaged with the game as we introduced a gender-neutral aesthetic, but the learning was still limited to sequencing simple movements rather than more advanced programming concepts (e.g., conditionals). Detailed issues are described below in terms of the focal point of our design and the results of the prototyping session.

Design Focus

Several research studies have asserted the need for gender equity in designing instructional materials for computer science education (Barab et al., 2005; Grover & Pea, 2013; Justice & Markus, 2010). With this need in mind, we considered several gender-neutral themes to improve the appeal and aesthetics of the game. Our final selection was "bears on an adventure." In line with the theme, colored areas in the previous prototypes were replaced with water, grass, and pit holes. Rewards (honeycombs, berries, fish) and obstacles (rocks and trees) were put on the board with physical objects made of playdough and small toys to provide a more interactive and tangible experience for the players.

With regard to the *game,* the primary goal was for players to randomly collect tiles of codes and assemble them on the coding board to move their characters on the action board. On the action board, players could earn points by collecting rewards, where each reward type had a different point value. The player with the most points in total would win the round.

On the coding board, the tiles were categorized and color-coded by words and non-words. To encourage *learning,* the block of code at the center was usable to both players; each round, a player could add to it or modify it to their liking. For example, in Figure 2.4, the first player constructed a block of code, "Turn [diagonal]" then "Move 2 [squares]." Building on existing blocks of code encouraged players to actively make strategies that could build into more advanced programming. Also, the player that used up their tiles first would end the round and receive bonus points. The goal of this reward was to encourage players to build more advanced codes so that they could use up their tiles more quickly.

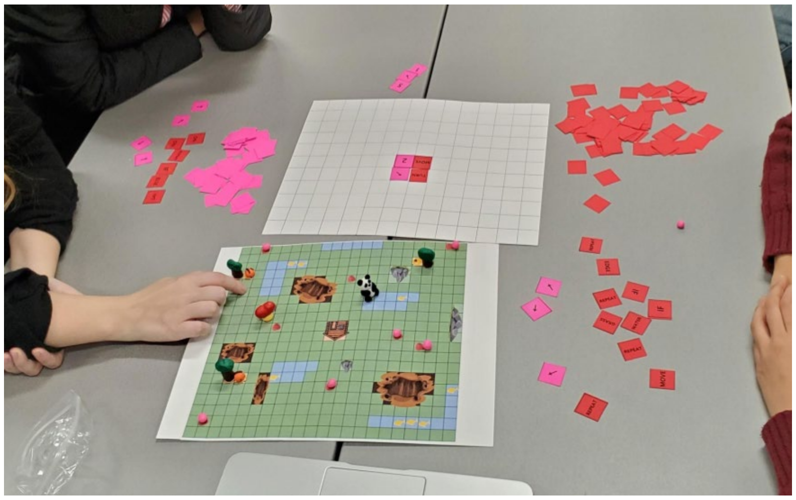


Figure 2.4: Graduate students playtesting prototype 3, with the tiles categorized and color-coded into two types: words and non-words/symbols.

Results

The prototype was playtested with five graduate students, including three novice players and two returning players. It was clear that the added aesthetics and story attracted the players' attention and increased their motivation to be more engaged in the game. The participants reported that the game was fun to play and that having physical objects on the board to collect or avoid made the game much more interactive.

However, now that the game had more solid rules and content, several significant limitations were more evident:

The players were confused over the different tile types. For example, the tile "move" was to be used with numbers to mean 'move [number] steps forward.' Similarly, the "turn" was to be used with an arrow to indicate 'turn to face [direction of the arrow].' The variation made gameplay confusing for the players, who needed repeated reminders about how to use the tiles on the coding board to move their character on the action board.

There was still an issue with encouraging players to engage in higher-level computer programming concepts such as conditionals and loops. One primary reason for this issue was that the gameplay required players to wait until they collected all the tiles needed to complete a complex command. This was frustrating and tedious. For example, it might take multiple rounds of gameplay before an if-then statement could be constructed to completion. Often, the need for the statement passed before the tiles could be played. As a result, players found it easier to rely on simple movements rather than attempt more advanced gameplay.

Prototype 4: Adding Scaffolds

Design Focus

Prototype 4 (see Figure 2.5) addressed the two issues that were revealed in the previous prototype. First, we added structure that reduced player confusion over the meaning of each tile. Second, we continued improving the gameplay to encourage players to engage in higher-level computer programming concepts. Playtesting revealed that both the graduate and the 5th-grade students found the game intuitive and were more motivated to use advanced codes than in previous versions of the game. One challenge arose regarding the retention of players' engagement and motivation. Detailed issues are described below in terms of the focal point of our design and the results of the prototyping session.

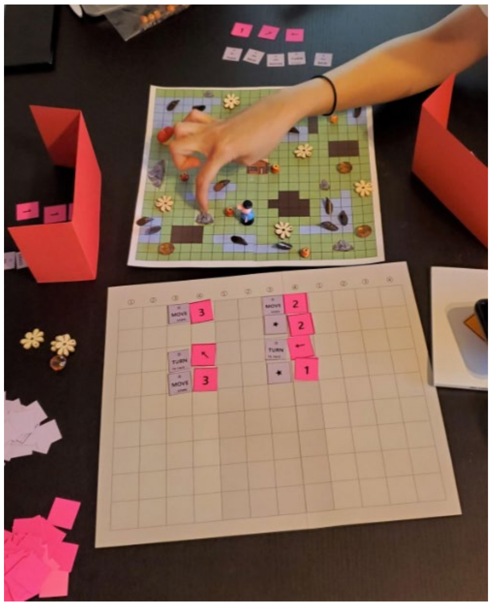


Figure 2.5: The final prototype. Numbers were added to the tiles and coding board to guide the placement of each tile.

Results

The prototype was played with two groups of playtesters: (1) a group of three adult players, including two novices and one expert in computer programming, and (2) one with a fifth-grade student who was familiar with working with Scratch. After one round of gameplay (20 minutes) for each group, all players agreed that they were motivated to use the advanced programming concepts. When the novice players played the game, they began by using simple movement codes. As the game progressed, they soon began creating more complex codes that involved conditionals and loops. Although returning players began using advanced codes sooner than novices, both groups created a similar level of statements by the end of each round. All playtesters explained that they used thinking processes similar to those needed to program on the computer. It was clear that the added scaffoldings and the revised reward system helped make the game more intuitive while achieving the learning goal.

One common issue experienced by both the graduate and 5th-grade learners was that interest in the game diminished over time. For both audiences, the first 10 minutes of gameplay were exciting and intriguing. They enjoyed thinking through different possible moves to earn rewards and points. The 5th-grade learner intentionally used an if-then conditional early in the game that allowed him to collect points more easily as the game progressed. However, once those points were earned after several rounds, he slowly began to lose interest in the game. The adult players similarly lost interest in gameplay once they began accumulating points, mainly because they were repeating the same gameplay pattern, and it was unclear what it would take to end the game and win it.

Both sets of playtesters recommended placing a time limit on the game so that there was pressure to earn the most points in a short period of time. This would make the win-state of the game more achievable for players and encourage them to continue finding ways to earn the most points possible, potentially sustaining interest as the game progressed. Another suggestion for improving and sustaining interest in the game was to add cards that would introduce fun but unpredictable challenges or rewards. Examples included allowing players to lose a turn, move to a random space on the game board, steal points from one another, or interfere with an opponent's current strategy.

Reflection

The purpose of this paper was to present our process of prototyping an educational game that sought to teach novice programmers the fundamental components of block-based programming in a Scratch environment. After four rounds of prototyping with both adults and a young learner, we were able to improve the various elements of the game core (Kalmpourtzis, 2018), including the *gameplay*, the *learning*, and the appeal of the game to the *players*. Each prototype resulted in new challenges that not only required design decisions but also helped balance the core elements while meeting our goals for the game. In the following sections, we reflect on our design process, highlighting three main areas in which our design process offers implications for the design of educational board games. The first focuses on the importance of having clear learning objectives when making decisions about the game. The second focuses on balancing learning with the goal of making the game fun for the players. The third focuses on developing in-game support that encourages players to engage in higher levels of computer programming concepts. Detailed explanations are presented below.

GAME: Making Design Decisions: Having clearly stated goals and objectives to help make critical decisions about the design of the game

Our game design originated with a clear learning objective; we wanted our players to construct and understand both basic and advanced computer programming skills through gameplay. This clarity proved to be an essential component of our decision-making throughout the game development process. To begin, it helped us decide which language and concepts to include for our target audience. We had several coding languages and tools to choose from, such as Java, Python, Scratch, and C++, as well as an option to make up our own language that taught coding concepts. Although those languages included similar programming concepts, the terms used in each language were distinct from one another. We ultimately chose to mirror Scratch's block-based programming environment because Scratch is widely used by K-5 teachers as an introductory platform for computer programming. By focusing on block-based programming as our primary learning objective, we were able to refine gameplay and mechanics with confidence. It helped us determine which programming terms to include on game tiles and which programming concepts to focus on in our game.

Having a clear objective also helped us make decisions about how to improve the game from one prototype to the next. In some cases, this led to improvements in the game design. Our use of two game boards closely mirrors the Scratch environment in which the coding space is kept separate from the animation area where those codes are executed. In other cases, our clear objective helped us to determine what *not* to include in our design. For example, playtesters repeatedly suggested that we make the game more exciting by allowing players to attack their opponents. In response, we prototyped a game dynamic that would allow players to attack one another. While we were able to make the attack function work to some extent, it ultimately resulted in gameplay that de-emphasized our learning goal. Rather than attending to the coding component of the game, players instead focused on finding tiles for attacking. We ultimately had to limit the attack function and create other mechanisms that improved gameplay while also achieving our learning goal.

Goal and Design Reflection. Having a clear learning objective helps designers make critical design decisions about what elements and mechanisms to include in the game and to which extent they should accept the various feedback from players.

PLAYERS: What Do Players Want? Using low-fidelity prototypes to receive honest feedback

Educational game designers are challenged to find the balance between making the game effective for learning while also appealing to the learners who play the game (Acosta & Denham, 2018; Greeff et al., 2017; Hopkins & Roberts, 2015; Kalmpourtzis, 2018). In fact, one of the most critical internal motivations for students to play educational games is to have fun (Long, 2007). This implies the importance of knowing the players' gameplay experience and what *they* want in a game to have fun, rather than solely focusing on the educators' goals. Early and frequent prototyping is essential in getting honest feedback from the players (Kelley & Kelley, 2013; Moggridge & Atkinson, 2007).

Our low-fidelity approach during initial prototyping helped us learn more about what players wanted without expending valuable resources. Our earliest prototypes were made using scraps of paper, small toys, and post-it notes that we could find in the room. This approach made it clear to playtesters that our prototype was in very initial draft form and need of tremendous improvement. As a result, they could easily understand the concept and intent of the game while being less hesitant to provide straightforward comments when asked to help us improve the prototype. Such feedback immensely helped in learning the players' preferences without expending our own resources needlessly. It also helped us, the designers, remain emotionally removed from the feedback we were receiving. We intentionally limited the time and effort we invested in creating the prototype. As a result, it was much easier to hear critical yet necessary feedback that helped us quickly improve and test new game components in the next prototype.

Goal and Design Reflection. Using low fidelity prototypes during earlier stages of game development can help designers more easily and quickly determine what the players want to do in the game environment.

LEARNING: Engaging learners in higher levels of concepts: Use of immediate feedback and exclusive rewards

One persistent design challenge had to do with creating opportunities for players to engage in higher levels of programming such as conditionals and loops, rather than being limited to just simple movements. We wanted to encourage players to use such advanced functions but at the same time make the game accessible to both novice and returning players. As revealed in our earlier prototypes, this was difficult to achieve. Most players focused on simpler movements because these were easier to employ and often guaranteed small but immediate rewards. Even when players could earn the largest points at the end of the game for using advanced concepts, the effect was minimal.

One of the major reasons for this challenge was a lack of immediate rewards for using advanced programming concepts. The timing of feedback is a design focus in video game settings. Delayed feedback can be motivating for players who are winning but at the same time discouraging for players who are losing (Turkay et al., 2014). The same idea can be applied to the board game setting. Even though the players in our game could receive the largest points at the end of each round for using advanced concepts, many players were more motivated to use simpler codes instead. This was especially true for players who were not in the lead; simpler codes allowed them to collect immediate rewards more quickly than waiting until the end. Thus, the solution we integrated into Prototype 4 was to provide instant rewards exclusively earned by using conditionals. This additional game mechanic was highly effective in encouraging players to be more actively engaged with using higher-level programming concepts as part of their gameplay.

Goal and Design Reflection. Providing immediate and exclusive rewards in a game can motivate players to engage with higher-level thinking.

Conclusion

Games are an increasingly popular mechanism for learning. In the context of computer science, they can help beginning programmers engage in computational thinking and acquire computer programming skills prior to working on a computer. This paper, then, provides greater insight into game development and the mechanics that can support learning computer science through gameplay. It is our hope that this design case serves as a foundation for educators and scholars to build upon as we move closer to the goal of improving computer science in K-12 settings.

References

Acosta, M. M., & Denham, A. R. (2018). Simulating oppression: Digital gaming, race and the education of African American children. *The Urban Review*, 50(3), 345-362. https://doi.org/10.1007/s11256-017-0436-7

Apostolellis, P., Stewart, M., Frisina, C., & Kafura, D. (2014, June). RaBit EscAPE: a board game for computational thinking. In *Proceedings of the 2014 conference on Interaction design and children* (pp. 349-352). https://doi.org/10.1145/2593968.2610489

Barab, S. A., Gresalfi, M., & Ingram-Goble, A. (2010). Transformational play: Using games to position person, content, and context. *Educational researcher*, *39*(7), 525-536. https://doi.org/10.3102/0013189X10386593

Barata, G., Gama, S., Jorge, J., & Gonçalves, D. (2013, October). Improving participation and learning with gamification. In *Proceedings of the First International Conference on gameful design, research, and applications* (pp. 10-17).

Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work?. In *Adventures between lower bounds and higher altitudes* (pp. 497-521). Springer, Cham. https://doi.org/10.1007/978-3-319-98355-4_29

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20-29. https://purehost.bath.ac.uk/ws/files/214932627/NZJACIT_Unplugged.pdf

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, 72, 145–157. https://doi-org.proxy-remote.galib.uga.edu/10.1016/j.compedu.2013.10.020

Computer Science Teachers Association (2017). *CSTA K-12 Computer Science Standards, Revised 2017*. http://www.csteachers.org/standards

Ericson, B., Adrion, W. R., Fall, R., & Guzdial, M. (2016). State-based progress towards computer science for all. *ACM Inroads*, *7*(4), 57-60. https://doi.org/10.1145/2994607

Fotaris, P., Mastoras, T., Leinfellner, R., & Rosunally, Y. (2016). Climbing up the Leaderboard: An Empirical Study of Applying Gamification Techniques to a Computer Programming Class. *Electronic Journal of e-learning*, *14*(2), 94-110. https://eric.ed.gov/?id=EJ1101229

Gibson, B., & Bell, T. (2013, November). Evaluation of games for teaching computer science. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 51-60). https://doi.org/10.1145/2532748.2532751

Greeff, J., Heymann, R., Heymann, M., & Heymann, C. (2017, September). Codebreakers: Designing and developing a serious game for the teaching of Information Theory. In *International Conference on Web-Based Learning* (pp. 13-22). Springer, Cham. https://doi.org/10.1007/978-3-319-66733-1_2

Gresse von Wangenheim, C., Araújo e Silva de Medeiros, G., Missfeldt Filho, R., Petri, G., da Cruz Pinheiro, F., Ferreira, M.N.F., Hauck, J.C.R. (2019). SplashCode--A Board Game for Learning an Understanding of Algorithms in Middle School. *Informatics in Education*, *18*(2), 259-280. https://files.eric.ed.gov/fulltext/EJ1233582.pdf

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1), 38-43. https://doi.org/10.3102/0013189X12463051

Hinebaugh, J. P. (2009). *A board game education*. R&L Education.

Hopkins, I., & Roberts, D. (2015). 'Chocolate-covered Broccoli'? Games and the Teaching of

Literature. *Changing English*, *22*(2), 222-236. https://doi.org/10.1080/1358684X.2015.1022508

Justice, S., & Markus, S. (2010). Educators, gender equity and making: Opportunities and obstacles. https://www.academia.edu/download/39055089/Justice-Markus-2015.pdf

Kalmpourtzis, G. (2018). *Educational game design fundamentals: a journey to creating intrinsically motivating learning experiences*. CRC Press.

Kelley, T., & Kelley, D. (2013). *Creative confidence: Unleashing the creative potential within us all*. Currency. https://pdflake.com/wp-content/uploads/2021/05/Creative-Confidence-PDF-Book-In-English-By-Tom-and-David-Kelley.pdf

Long, J. (2007). Just For Fun: using programming games in software programming training and education. *Journal of Information Technology Education: Research*, *6*(1), 279-290. https://www.learntechlib.org/p/111422/.

Mcgettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., & Mander, K. (2005). Grand challenges in computing: Education—a summary. *The Computer Journal*, *48*(1), 42-48. https://doi.org/10.1093/comjnl/bxh064

Moggridge, B., & Atkinson, B. (2007). *Designing interactions* (Vol. 17). Cambridge, MA: MIT press.

Nah, F. F. H., Zeng, Q., Telaprolu, V. R., Ayyappa, A. P., & Eschenbrenner, B. (2014, June). Gamification of education: a review of literature. In *International conference on hci in business* (pp. 401-409). Springer, Cham. https://doi.org/10.1007/978-3-319-07293-7_39

Olsson, M., Mozelius, P., & Collin, J. (2015). Visualisation and Gamification of eLearning and Programming Education. *Electronic journal of e-learning*, *13*(6), pp452-465. https://www.academic-publishing.org/index.php/ejel/article/download/1947/1910

Piteira, M., & Costa, C. (2012, June). Computer programming and novice programmers. In *Proceedings of the Workshop on Information Systems and Design of Communication* (pp. 51-53). https://doi.org/10.1145/2311917.2311927

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, *13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

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

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

Thompson, D., Bell, T., Andreae, P., & Robins, A. (2013, March). The role of teachers in implementing curriculum changes. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 245-250). https://doi.org/10.1145/2445196.2445272

Tsan, J., Lynch, C. F., & Boyer, K. E. (2018). "Alright, what do we need?": A study of young coders' collaborative dialogue*. International Journal of Child-Computer Interaction*, 17, 61–71. https://doi-org.proxy-remote.galib.uga.edu/10.1016/j.ijcci.2018.03.001

Tsarava, K., Moeller, K., & Ninaus, M. (2018). Training computational thinking through board games: The case of Crabs & Turtles. *International Journal of Serious Games*, *5*(2), 25-44. https://doi.org/10.17083/ijsg.v5i2.248

Turkay et al. (2014). Toward understanding the potential of games for learning: Learning theory, game design characteristics, and situating video games in classrooms. *Computers in the*

Schools, 31(1-2), 2-22. https://doi.org/10.1080/07380569.2014.890879

Walton-Hadlock, M. (2008). Tots to tweens: Age-appropriate technology programming for kids. *Children & Libraries*, *6*(3), 52. https://www.proquest.com/scholarly-journals/tots-tweens-age-appropriate-technology/docview/212174009/se-2?accountid=14537

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wu, S. Y. (2018). The Development and Challenges of Computational Thinking Board Games. In *2018 1st International Cognitive Cities Conference (IC3)* (pp. 129-131). IEEE. https://doi.org/10.1109/IC3.2018.00-45

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In *Emerging research, practice, and policy on computational thinking* (pp. 205-220). Springer, Cham. https://doi.org/10.1007/978-3-319-52691-1_13

APPENDIX

Appendix A

Summary of Challenges by Prototype

Phase	Game core			
	Game	Players	Learning	
Initial Goal	Make the game fun to play	Create the game core that appeals to players	Teach block-based programming (i.e. Scratch)	Engage in concepts be-yond movement-based gameplay
1	Game mechanics (tiles) were too complex; also, more space was needed to assemble codes on the game board.	The placeholders in the game were difficult to understand for novice players. The game was playable but not exciting or appealing.	The coding mechanics used in the game were significantly different from Scratch,	The advanced programming concepts in the game were too confusing.
2	Added a second game board and simplified tiles/ codes, but the game was not very exciting.	The placeholders were removed, and the gameplay mechanic simplified.	Tiles simulated how blocks function in Scratch, and the concepts used in the game aligned with those in Scratch	The simplified mechanic made the game more fun but did not encourage more than simple movements.
3	Game mechanics supported the assembling of codes in blocks, Aesthetics made the game more interesting.	The game became more appealing, but the language and use of tiles with advanced concepts was still confusing.		Gameplay improved but focused on simple movements due to the mechanics around using tiles with advanced concepts.

4	Rewards were added to promote engagement, but the game still needed more fun elements.	Scaffolds were added to make the game more intuitive and easier to understand, but the players' interest diminished over time.		Bigger and exclusive rewards were added that encouraged conditionals; this motivated the players to use more complex codes.
Next Steps	Limit the duration of a single round; increase random elements to make gameplay more exciting and challenging.		Improve ways to engage players in advanced concepts such as loops and variables.	

CHAPTER 3

USING A BOARD GAME FOR COMPUTER PROGRAMMING EDUCATION: A QUALITATIVE EXPLORATORY STUDY[2]

[2] Yang, D., & Kopcha, T. Submitted to *Computer Science Education*

Abstract

Background and Context: Computer programming is increasingly essential in K-12 education for developing problem-solving skills. This exploratory study explores the efficacy of the board game *Lucky Codes* in teaching programming concepts to elementary students through unplugged learning methods.

Objective: To evaluate how *Lucky Codes* supports the development of programming skills, such as sequencing and conditionals, among elementary school students during gameplay.

Method: A qualitative approach was used, involving video recordings and multimodal transcription of gameplay sessions with four elementary students. This allowed for a detailed analysis of interactions and strategies.

Findings: The game effectively facilitated the learning of programming skills. Students demonstrated meaningful engagement with key programming concepts like sequencing and the use of conditionals to progress through the game levels.

Implications: *Lucky Codes* showcases the potential of board games as effective pedagogical tools for foundational programming education. The findings support integrating such unplugged activities into the curriculum to make programming accessible and engaging for young learners and enhance their computational thinking skills.

Introduction

Computer programming has become increasingly important in K-12 education. Programming is largely viewed as a way to teach essential problem-solving skills and build foundational computing concepts like decomposition, abstraction, sequencing, conditionals, algorithms, and loops (Cuny et al., 2010; Ezeamuzie & Leung, 2022; Grover & Pea, 2018; Wing, 2006). Whereas programming was once seen as a domain for computer scientists, it is now recognized for the way it encourages students to develop analytical abilities; as such, leading educators around the world now advocate for programming to be included in the K-12 curriculum (Lindberg et al., 2019; Nambiar, 2020; Wu et al., 2020). A variety of teaching strategies have been introduced to convey these programming concepts. For example, visual block-based programming tools such as Scratch, Alice, Code.org, and Kodu were developed to help novice students learn to program more easily using everyday language without having to learn complex programming syntax. While these tools have been shown to help enhance children's computer programming skills, scholars have found that hands-on experiences can offer invaluable insights before moving into screen-based programming. For example, Sun et al. (2021) suggested employing instructional techniques like hands-on programming practice (e.g., card sorting activity), which can help students understand the difficult terminology in programming and enhance students' interest in computer education. Other scholars have reported that hands-on, unplugged learning activities can provide a much deeper level of engagement (Bell et al., 2012) and effectively improve learners' perceptions and interest in computer science (Taub et al., 2012).

To that end, unplugged activities have gained traction as a pedagogical approach to introduce computer programming skills to children because they eliminate the need for computer or digital devices and provide tangible and physical objects to develop more concrete ideas about abstract programming concepts (e.g., variables, conditionals) (Kotsopoulos et al., 2017; Pecher & Zwaan, 2005). Such activities can be beneficial in K-12 contexts in that they make learning more accessible to a broader audience because they do not require the technological resources that come with programming on a computer (Bell & Vahrenhold, 2018; Gibson & Bell, 2013). Board games have emerged as a compelling avenue for introducing children to computer programming through a tangible, hands-on approach. They offer a playful and interactive environment that is conducive to the exploration of core programming skills, including problem-solving, algorithmic thinking, sequencing, and debugging (Berland & Duncan, 2016; Tsarava et al., 2018).

While several board games exist to teach programming skills to children, such as Robot Turtles, Code Master, and CodeMonkey Island, many primarily emphasize basic path-finding mechanisms, neglecting more advanced programming concepts like conditionals and loops (Poole et al., 2022; Wu et al., 2020). Moreover, some games overly prioritize mechanics involving card stacking, where players stack cards with specific instructions or actions to achieve a particular outcome within the game; such game mechanics can create a disconnect between the game's mechanics and the actual programming environment (Wu et al., 2020). Research in this domain has predominantly relied on quantitatively measured self-reported data or user experience evaluations within the game, overlooking the crucial aspect of how learning unfolds during gameplay (Gresse von Wangenheim et al., 2019; Kuo & Hsu, 2020; Lee et al., 2020). Fewer studies have explored the relationship between student interactions and game mechanics that facilitate learning (Poole et al., 2022).

Addressing these observed gaps, our team developed *Lucky Codes,* a board game devised to enrich students' programming skills while reflecting the Scratch programming environment (Yang & Kopcha, 2022). *Lucky Codes* challenges students to employ programming skills such as decomposition, sequencing, and conditional thinking to guide a game character through a board scattered with coins, where the goal is to collect a certain number of coins before your opponent. This study explores how different board game mechanics in *Lucky Codes* supported the programming skills of four elementary school children during gameplay. Using qualitative methods, we video-recorded the children as they played the game in pairs over two 10-minute rounds of gameplay. We then used a multimodal approach to develop and analyze transcripts that contained their game moves paired with their conversations during gameplay. Our analysis focused on the way different game mechanics were associated with programming skills as they related to specific moves, as well as how different concepts of computer programming developed over multiple rounds of gameplay. The research questions guiding this study were: (1) How did the game mechanics support children's programming skills during each turn of the gameplay? and (2) How did the presence of the children's programming skills develop or change over time? In addressing these questions, this paper sheds light on the educational potential of board games like *Lucky Codes* in enhancing children's programming skills, laying a foundation for subsequent research prospects.

Game Description: *Lucky Codes*

Lucky Codes is an educational coding board game designed for elementary school students in grades 3 to 6. It was developed by the authors as part of a design-based research project (Yang & Kopcha, 2022) with the aim of blending fundamental programming concepts with advanced ones, making it an engaging and educational experience. Figure 3.1 illustrates the components of *Lucky Codes*, and Figure 3.2 displays the photo of students playing the game with descriptions of the gameplay.

The game was designed to resemble the block-based programming environment Scratch, which appeals to both novice and experienced players. It uses two game boards to support gameplay, consisting of one board where the students assemble tiles into a sequence of codes and another where the game character performs the actions reflected in the code, as one would do in Scratch. This two-board system was implemented to help bridge the gap between the physical and digital realms of computer programming education (Wu, 2018).

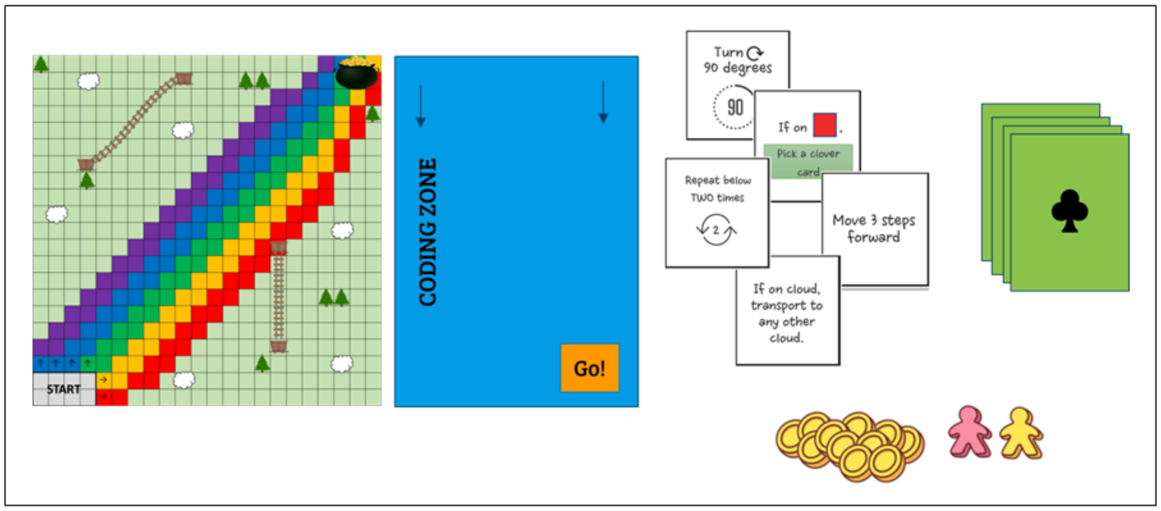


Figure 3.1: The components of *Lucky Codes*. (From left) action board, coding board, code tiles, clover cards, coins, and characters.

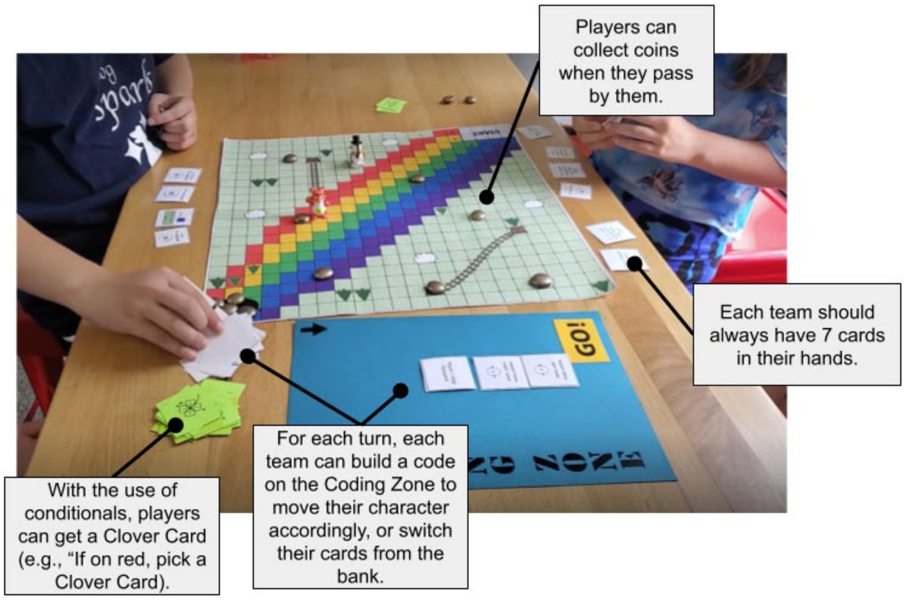

Players can collect coins when they pass by them.
Each team should always have 7 cards in their hands.
For each turn, each team can build a code on the Coding Zone to move their character accordingly, or switch their cards from the bank.
With the use of conditionals, players can get a Clover Card (e.g., "If on red, pick a Clover Card).
GO!

Figure 3.2: Screenshot of students playing *Lucky Codes* with descriptions of gameplay.

The intent of this design was to help students understand and engage with the similar cognitive processes they would use in a digital programming environment, such as planning, executing, and revising code based on visual feedback. This could potentially promote an easier transition between physical board gameplay and block-based digital programming environments (Hajian, 2019). The game also mirrors Scratch's terminology, including actions like "move _ steps," "turn [direction] _ degrees," and "repeat," and code assembly approach. Players connect the code tiles horizontally to establish the sequence of actions. Loops are introduced at the start of the corresponding sequence, mirroring the mechanic of Scratch. In *Lucky Code's* code tiles, however, modifiers have been added to the verbs (e.g., "move 1 step 'forward,'" "repeat 'below' 2 'times'") to help players more easily understand what each code tile means. Figure 3.3 provides a comparison between the code blocks in Scratch and the code tiles in *Lucky Codes* for executing the same sequence of actions.

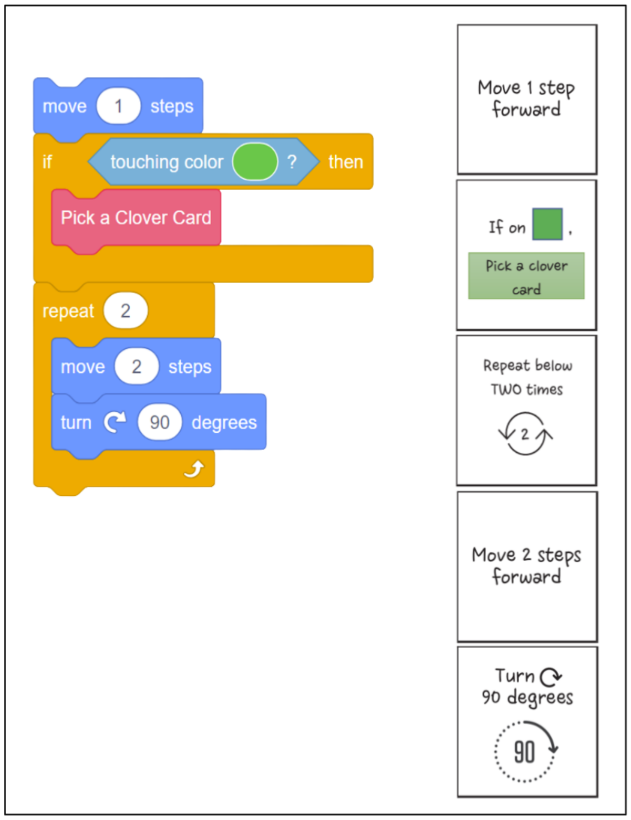


Figure 3.3: A comparison: scratch code blocks (left) vs. *Lucky Codes* code tiles (right) for executing the same sequence of actions.

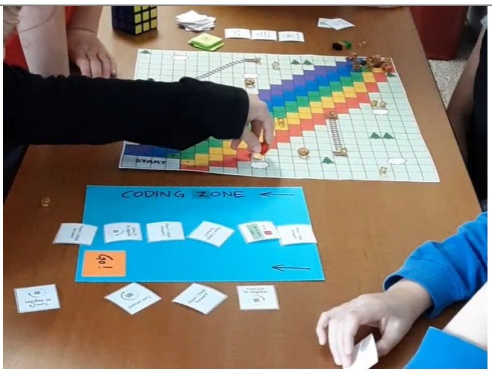
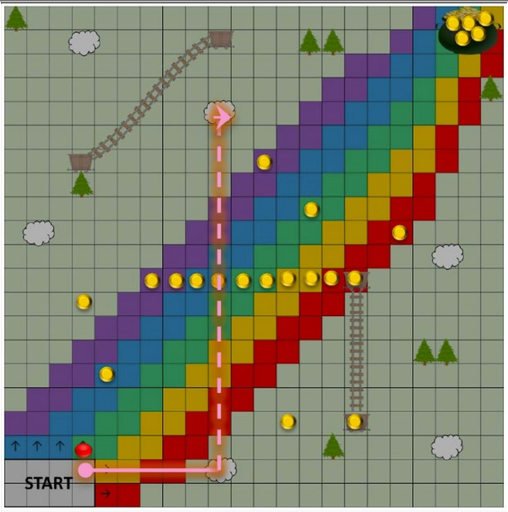
To play the game, students are paired up to form teams, with each team competing against another. Each team places an equal number of coins on the board as they see fit and picks seven random code tiles. These tiles encompass basic movements, conditionals, and loops, with instructions like “Move 3 steps forward,” “Turn 90 degrees in any direction,” “If on green, pick a Clover Card,” and “Repeat below 2 times.” "Clover Cards" introduce an element of unpredictability: they are cards that players can acquire by using color-conditionals, such as “If on red, pick a Clover Card.” These cards can either assist or challenge players, presenting commands like "Follow the code made by the other team" or "Move two steps in any direction." This feature, adding a layer of chance, was incorporated to boost both the educational aspect of the game and its entertainment value, as unpredictable elements can heighten engagement and motivation in educational games (Kalmpourtzis, 2018).

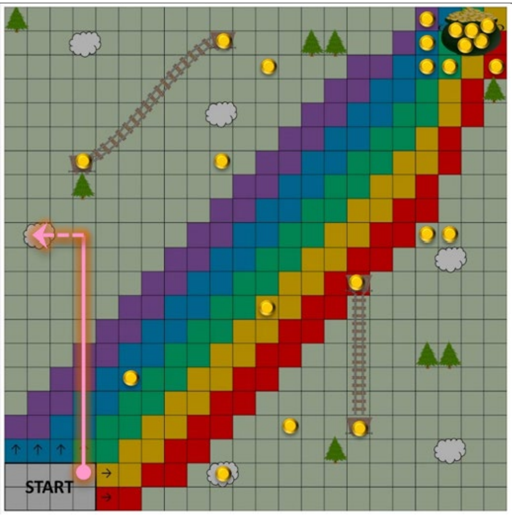
The goal of the game is to collect more coins than the opponent team within a given time by reaching the pot of gold at the end of the rainbow (where players can get five coins at once) or quickly moving around the board to collect five of the scattered coins. During each turn, teams can either formulate a code to move their characters or draw one to two new code tiles from the bank. They also have the option to incorporate any portion of their opponent's previously used codes into their own sequence. Players assemble their code sequences in the designated Coding Zone. To execute their move, they tap the “Go!” button located at the Coding Zone's bottom, which finalizes their codes. The players then move their characters on the main game board according to the formulated code.

Lucky Codes' game mechanics have been refined through iterative design-based research, blending fundamental programming concepts like sequencing and goal decomposition with more advanced concepts like conditionals and loops (refer to Yang & Kopcha (2022) for a detailed design story on the game). These concepts were selected because they represent the foundational concepts associated with teaching elementary-aged children about computer programming (Bers, 2019). Table 3.1 showcases the various mechanics in *Lucky Codes*, their theoretical underpinnings, and examples of their enactment in the game. For example, central to the gameplay is the task of assembling code tiles in the Coding Zone to create a sequence of movements that the game character will execute on the game board. This dynamic supports the essence of algorithmic thinking in programming in that children can see what happens when they create "precise step-by-step plans or procedures to meet an end goal or to solve a problem" (Grover & Pea, 2018, p. 24).

The game was improved from its previous version (see Yang & Kopcha, 2022) in an effort to reward players for using the underlying concept of a conditional to advance gameplay. This was accomplished by introducing colored elements on the board that invite players to use a color-conditional tile to attain a Clover Card. For example, players can use a tile like, "If on red, pick a clover card," to draw a random chance card that can potentially benefit their gameplay. This design choice serves two purposes. First, it encourages players to use conditionals more frequently, discouraging an over-reliance on basic movement commands. Second, it introduces randomness to the type of benefit they might receive. Each Clover Card has a different purpose; some allow the player to move further, while others allow them to steal coins from the opponent or even change the opponent's position on the game board. This randomness meant that players could not fully anticipate the nature of the Clover Card ahead of time, which can heighten engagement and motivation in educational games (Kalmpourtzis, 2018). Instead, they had to keep gathering them until they found one that benefited them most. Another way that the use of conditionals was incentivized was the introduction of a cloud-based teleportation feature. With the conditional, "If on a cloud, transport to another cloud," players have the strategic advantage of moving a larger distance swiftly, further promoting the use of conditionals during gameplay.

Table 3.1: Key game mechanics, corresponding theoretical foundations, and gameplay excerpts (next page).

Game Mechanic	Theoretical Foundation	Gameplay Excerpts
Assembling code tiles to create a sequence of characters' movements on the board	Sequencing is a precise, step-by-step procedure to solve a problem (Grover & Pea, 2018).	 Students place code tiles sequentially in the Coding Zone while their characters perform corresponding movements on the game board. The Coding Zone was designed so that students could observe the moves of the other team and re-use tiles that were helpful to their own strategy. Student: “Should we do what they are doing? Just put all the cards in [one shared] pile? So, like, the [tiles] are just, there?”
Shared coding space	Social learning suggests that human behavior is acquired through observation and modeling, which provides insight into how new behaviors are carried out; this serves as a reference for their own actions in the future (Bandura & Walters, 1977).	
Breaking down a problem into a smaller number of steps	Decomposition is breaking a problem into smaller subproblems to make the problem more approachable and manageable (Grover & Pea, 2018).	 Student: “I'm gonna try to get to that cloud and then use this [cloud-conditional card] to get all the way up [to the upper cloud]. All right. So, move forward one step. Then two steps would be here, and then three steps. Then, ‘If on a cloud, then transfer to another cloud.’ I would get that [coin], and we'd be [on the upper cloud].”

“If-then” statement tiles	Conditionals involve the use of if-then constructs and are fundamental to both computational thinking (Grover & Pea, 2013, 2018) and basic programming (Shute et al., 2017).	 Codes: Repeat below 2 times / Move 2 steps forward / If on blue, pick a Clover Card / Move 3 steps forward / Turn 90 degrees Student: “... Alright, so one, two [steps]. If on blue. So, we'd get a clover card and then 1, 2, 3, and then one, two. And then, uh, if on a blue, which we aren’t [on blue]. [Move] 1, 2, 3… and then turn.”

Literature Review

Computer Programming Education in K-12

In 1980, Papert first championed the idea that learning computer programming could empower students with critical thinking skills transferable to other subjects like mathematics and physics. This notion laid the groundwork for emphasizing the pivotal role of computer programming education. Wing (2006) similarly argued that individual programming skills are valuable tools to help students grasp computational thinking. This perspective catalyzed the movement to provide computer programming courses to students at all levels in K-12 schools.

The global recognition of the significance of computer programming in education led to its inclusion in core curricula in many countries. England, for example, led the way in 2014 by mandating computer programming as a subject in the K-12 curriculum to nurture students' digital literacy (Wong et al., 2015). Finland soon followed suit, recognizing that the "future will be built by those who know how to code" (Haaramo, 2014). The United States launched "Computer Science for All" in 2016 to equip all American students with computer science skills, including coding (Vihavainen et al., 2011). Subsequently, several other countries, including Australia, Estonia, France, Israel, South Korea, China, Singapore, Taiwan, and Canada, integrated coding into their national school curricula (Linneberg & Korsgaard, 2019; Nambiar, 2020; Wu et al., 2020).

Computer programming is especially recognized in the K-12 context for teaching students important problem-solving skills used by computer scientists (Grover & Pea, 2018; Lye & Koh, 2014; Resnick et al., 2009; Selby, 2012; Wing, 2006, 2008). Computer programming challenges students to break down complex problems into manageable parts and foster a structured approach to finding solutions (Guzdial, 2008; Wing, 2006). It encourages logical reasoning and precision, which not only enhances their problem-solving skills but also has the potential to address other academic areas such as mathematics, science, social sciences, and language arts (Barr & Stephenson, 2011; Brennan & Resnick, 2012; Grover & Pea, 2013). As a result, students equipped with programming knowledge are better prepared for the increasingly digital world, where computational thinking is a valuable asset (Barr & Stephenson, 2011; Brennan & Resnick, 2012).

In elementary computer programming classes, several core concepts are commonly taught to provide students with a strong foundation in computational thinking (Barr &

Stephenson, 2011; Brennan & Resnick, 2012; Grover & Pea, 2013; Guzdial, 2008; Wing, 2006). These concepts include variables, which allow students to store and manipulate data; loops, which enable them to perform repetitive tasks efficiently; conditional statements, which introduce the concept of decision-making in code; and functions, which encourage modular and reusable code design. Additionally, students learn about algorithms, which are step-by-step procedures for solving problems, and decomposition, which helps break down large problems into smaller and more manageable chunks. These fundamental concepts not only empower students to create simple programs but also lay the groundwork for more advanced programming skills as they progress in their education (Barr & Stephenson, 2011; Brennan & Resnick, 2012; Grover & Pea, 2013; Guzdial, 2008).

Challenges of CT Education and the Role of Unplugged Games

Despite the recognized importance of teaching computer programming to K-12 students, multiple challenges still remain. Firstly, many programming concepts require a high level of abstraction, which makes learning challenging for novice students (Piteira & Costa, 2013; Rijke et al., 2018; Wong et al., 2015). This level of abstraction can contribute to a lack of student interest in programming (Kadar et al., 2021; Rahmat et al., 2012). Furthermore, some students perceive programming as challenging or "nerdy," affecting their engagement and willingness to persist in learning to program (Andersen et al., 2003, p. 1). Moreover, the availability of adequate technology, software, and teaching resources for programming instruction is often limited in certain schools, further hindering effective programming instruction (Kim et al., 2019; Sentence & Csizmadia, 2017). These challenges emphasize the need for targeted interventions and additional instructional support that can promote successful programming education for K-12 students.

To address these challenges, educators have turned to alternative instructional tools, with one of the most promising options being unplugged tabletop games. Game-based learning, in general, is well-known for its potential to promote the motivation and engagement of students (e.g., Kapp, 2012; Mayer, 2014; Plass et al., 2015; Plass et al., 2020). Games include features such as incentive systems, competition, instant feedback, and challenges, all of which promote motivation and engagement and are found to have a solid connection to student achievement (Shute et al., 2017). Another aspect of games that can encourage motivation is that they allow graceful failure (Plass et al., 2020). Players expect to have failures in the first few attempts when playing games. These failures create challenges that make gameplay more interesting, giving players a higher sense of achievement when they succeed. The low consequences of failure within a game environment encourage risk-taking, exploration, and experimentation (Hoffman & Nadelson, 2010). Mladenović et al. (2016) noted how games could effectively stimulate students' curiosity and appetite for challenge, making the learning of abstract programming concepts more appealing: "Students are not even aware they learn problem-solving, debugging, and making scenarios; instead, they think they are simply [playing]" (Mladenović et al., 2016, p. 523).

Furthermore, unplugged games offer tangible objects that can help students develop more concrete ideas of abstract coding concepts (Pecher & Zwaan, 2005). Research by Jiang et al. (2023) suggests how the tangible nature of board games can positively affect children's development of procedural knowledge for computer programming. The study's key finding emphasizes that children can master procedural knowledge more effectively in games that incorporate meta-gaming than in those that do not. For instance, the concept of "conditionals" is highly challenging for young children to grasp through formal definitions. A game environment can introduce the concept in a concrete, tangible manner in which children are guided to analyze, represent, abstract, and select conditions to solve a specific problem. In doing so, the concept of a conditional is imparted through the game mechanic without requiring the semantic knowledge needed to construct or program a conditional on a computer. This example illustrates how meta-gaming within unplugged games transcends language limitations and facilitates the understanding of complex programming concepts in a concrete way, providing valuable insights into the intricate connection between gaming experiences and skill development in programming education.

Recently, scholars who used unplugged board games to cultivate students' programming skills reported improved learning motivation of the participating students (Berland & Lee, 2011; Harris, 2008), which could serve as evidence of the effectiveness of unplugged games in programming education. Poole et al. (2022) reviewed 24 tabletop games related to programming education and concluded that such tabletop games could potentially increase interest in the subject for young students. However, they also emphasized that it is essential to understand how the game's design supports which types of programming skills for more purposeful use of the game. However, they also emphasized that much more research is needed to understand the effects of these games, including how students executing codes within games support learning and understanding of the intended programming concepts. Scirea and Valente (2020) also agreed that more work is needed to evaluate these programming-relevant board games and identify which game supports which skills.

Method

Participants and Procedure

Students from grades three to five were chosen for this study because several state standards began introducing more advanced programming concepts like conditionals and loops at this age range (e.g., Georgia Department of Education, 2022; Indiana Department of Education, 2023). The game *Lucky Codes* was designed specifically with this age group in mind. Out of the 12 students who were solicited for the study, four were selected based on their willingness to play the game and join the research. The recruitment began with researchers clearly explaining the study's goals and tasks. Those who were interested were given both parental permission and minor assent forms, which went over the details of the study and highlighted that participation was completely voluntary. In the following week, only students who handed back both signed forms and showed genuine excitement were included, ensuring informed consent and voluntary participation in line with ethical guidelines. This process resulted in a balanced group of two male and two female students; of these, three had no previous exposure to coding. The fourth had limited introductory experience with Scratch but had not previously worked with higher-level concepts like loops and conditionals.

The students played the game during their Makerspace class during the school day. The researcher explained the game rules, and the students played two rounds of the game, each lasting 10 minutes. In between the rounds, the players reset the game board layout to arrange the rewards differently. The session took about 30 minutes in total and was video recorded.

Data Collection

Data was collected by video-recording participants as they played two 10-minute rounds of the game. Video recording allowed us to set up the research environment such that both the gameboard and participants' thoughts about the gameplay could be captured simultaneously for transcription and analysis later. We selected this approach because our goal was to gain a fine-grained understanding of the moment-by-moment thinking that took place during gameplay. We wanted to capture both images and dialogue related to each move that the participants made so we could better understand how computational thinking emerged as part of their gameplay. The researchers were present during the gameplay sessions to provide guidance to the players, reminding them of game rules and ensuring they avoided major errors in coding or character movements. The students quickly grasped the game, requiring only minimal intervention, primarily at the start of the game.

Analysis

The analysis began with transcribing the video into a multimodal transcript. As described by Bezemer (2014), multimodal transcripts bring the visual and textual components of the participants' interactions together into a single account so that a researcher can infer the meaning behind those interactions. While there is no single way to generate a multimodal transcript, the overall goal is to capture both the verbal and non-verbal modes associated with how a phenomenon emerges over time among a group of people being observed (Bezemer, 2014; Cowan, 2014). In this study, we created a visual representation of the character's actions on the game board during each turn of each round of gameplay. The visual contained a miniature game board and arrows displaying the character's moves during each turn. Each visual was paired side-by-side with the tiles each team used during each turn of gameplay. Each visual was also paired with a word-for-word transcription of the participants' spoken words so that we could understand the thinking behind each move. Figure 3.4 displays a sample of our multimodal transcription associated with the first two turns of the first round of gameplay.

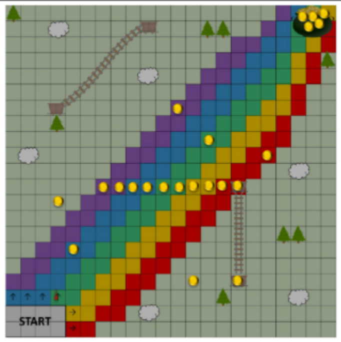
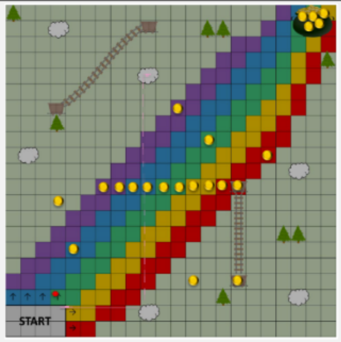
Round 1			
Team 1 – Move 1 00:05-01:30	Move 1 step – If on [green], pick a clover card: "Stand next to the other team" *<– decided to keep the card for later>*	- I think we should use both of these. Move one step. I know we should do that, right. Well, yeah. Cause then we can't go out. So yeah, we should definitely move one step. And uh, which one of these should we use? Which one of these should we… or does this count as a step? (- Yes, that's one step.) - Oh, okay. That means it can't be that or that [*If on blue/purple, pick a clover card]*. So which of these two, I guess this doesn't matter. Um, move, I'd say the green. - Yeah. That's it. Right, Toby? Go. - [Toby picks a clover card] "Your friend has summoned you. Choose one player and stand next to him." [Decides to play it later since they were the first move.]	
Team 2 – Move 1 02:30-03:22	Move 1 step – Move 2 steps – Move 3 steps – "If on a cloud, transport to any other cloud" *<Collected 1 coin>*	- Okay. Um, let's see. - (Team member joins from the restroom break.) Wait, what do we do? - Like just trust, trust. (- You're a team so you should discuss.) - So I'm gonna try to, **I'm gonna try to get to that cloud and then use this (card) to get all the way up here.** Oh, okay. All right. So move forward one step. Then two steps would be, … and then three steps. **Then on a, if on a cloud, then transfer to another cloud. I would get that, and we'd be there.** I will do that. [Presses "go".] Okay. - That's what I want. - You're welcome. And then I still have my "Move 90 degree" .	

Figure 3.4: A sample of the multimodal transcript.

When the multimodal transcripts were complete, the researchers (the authors of this paper) began coding the data. Coding the data meant that we assigned labels to specific parts of our transcripts that reflected both the programming concepts that were at play and the strategies that the teams employed. In this way, we used a combination of deductive and inductive coding, which is the most commonly used coding approach (Linneberg & Korsgaard, 2019). During the coding process, the researchers carefully reviewed the transcripts and identified sections that reflected the initial codes, highlighting them for further analysis and discussion. This collaborative approach ensured that the coding was thorough and well-considered, with agreement reached through ongoing dialogue and consensus-building. The details of the coding process are presented more fully in the following paragraphs.

Deductive Coding

We started with a deductive approach, which "ensure[s] structure and theoretical relevance from the start, while still enabling a closer inductive exploration of the deductive codes in later coding cycles" (Linneberg & Korsgaard, 2019, p. 264). Initial codes were drawn from the literature on computational thinking and computer programming, including algorithmic thinking, sequencing, decomposition, conditionals, and loops. The definitions of each code appear in Table 3.1. For example, we used Grover and Pea's (2018) definition of decomposition, which involves "breaking a problem down into smaller sub-problems to make the problem more tractable and the problem-solving process more manageable" (p. 27). Algorithmic thinking was defined as using "precise step-by-step plans or procedures to meet an end goal or to solve a problem" (Grover & Pea, 2018, p. 24). We read through the transcripts and highlighted parts that reflected the initial codes.

Inductive Coding

After our deductive coding, we used an inductive approach to maintain fidelity to the data (Linneberg & Korgaard, 2019) and to explore any elements in the data that might suggest a need to reconsider the existing theoretical framework (Pierce, 1978). As a result, new codes emerged from the data, including “problem-solving,” “collaboration,” “borrowed strategy (from the other team),” and “sense of achievement.” While identifying the codes, we captured images that suggested how each mechanic would support the skills in a move and conversation and arranged them side by side. We repeatedly reviewed the transcript to examine the data within a move and compare the data across different moves. We then looked at the multimodal transcript to establish a broader theme under which the codes fell. We focused our attention on recurring patterns of codes rather than isolated instances. This approach allowed us to identify consistent themes and patterns in the data across multiple moves and conversations. By using both deductive and inductive approaches to coding, we were able to stay true to the data while also incorporating established theoretical frameworks into our analysis.

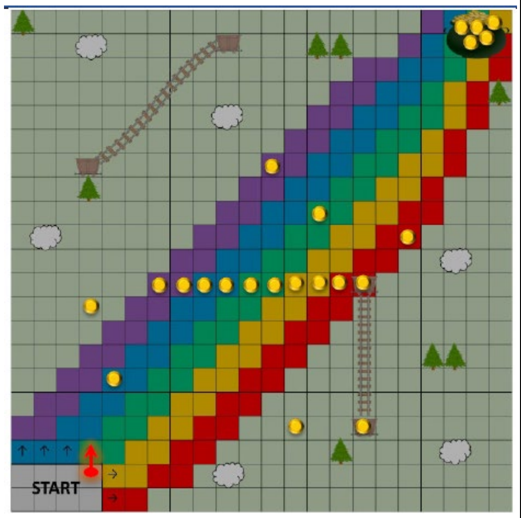
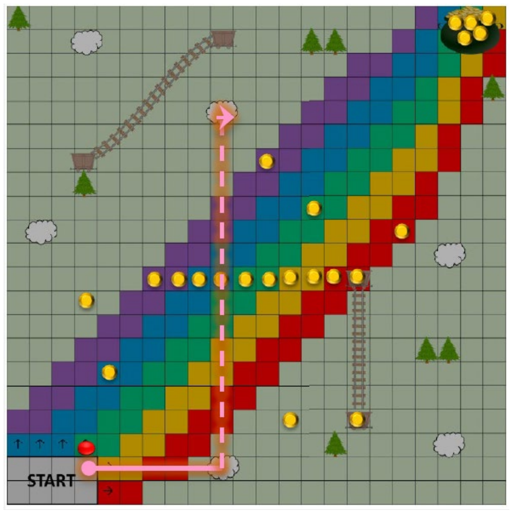
Results

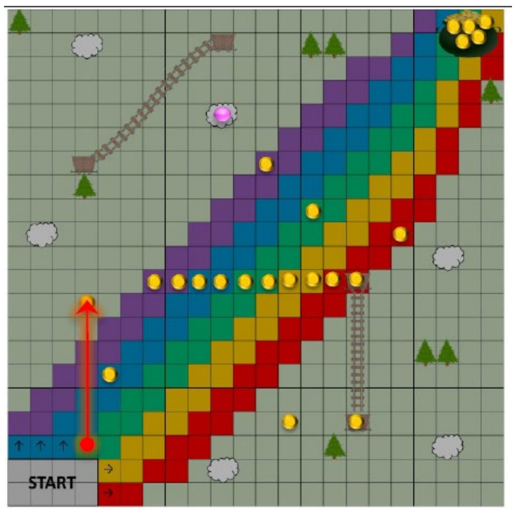
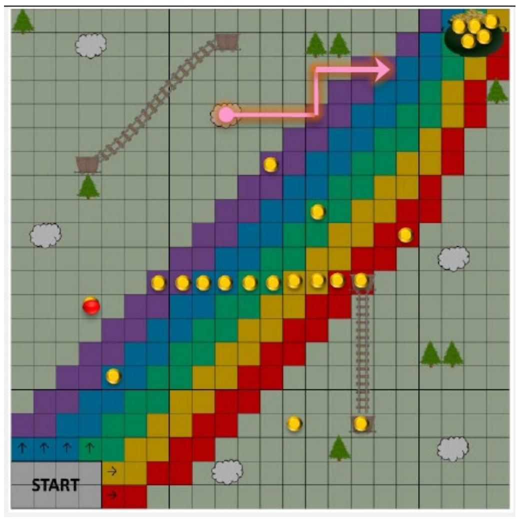
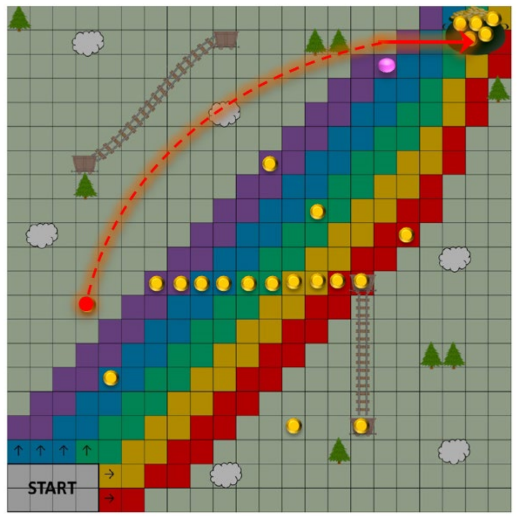
The game mechanics supported our participants' use of programming skills in three ways. First, the game required students to engage in both sequencing and decomposition. They analyzed the game board, broke down the goal of reaching the pot of coins into smaller sub-goals, and used game tiles to create sequences of commands moving their game pieces closer. As these actions were intertwined in gameplay, we combined sequencing and decomposition into a single theme. Second, the game supported the use of conditionals. In this study, conditionals were reflected any time the students used a tile or game card containing a pre-constructed "if-then" statement. There were two types of conditionals in *Lucky Codes* - a cloud-conditional that allowed players to jump from cloud to cloud and a color-conditional that awarded a clover card. Programming skills were also supported by the theme 'Learning from Each Other,' as teams adapted strategies observed from their opponents over time. These themes are described below as they unfolded over each round of gameplay.

Sequencing and Decomposition

The first programming skill that the game mechanics supported was sequencing and decomposition, which took place throughout both rounds of gameplay. Figure 3.5 displays the gameplay that took place in the first round of the game. Each team began by decomposing the larger goal of reaching the pot of coins at the far end of the game board. This process involved breaking the distance between the start and the end goal (i.e., the pot of coins on the far side) into smaller chunks, then sequencing a series of tiles that would move their game piece closer to their immediate goal. For example, they first set a goal of moving closer to a single coin near the center of the board, which they achieved in two turns. In their first turn (Team 1 - Turn 1), they moved forward one square. In the next turn (Team 1 - Turn 2), they moved forward to the single coin. As such, both teams consistently broke down the main objective and planned steps to reach interim goals.

Figure 3.5: Round 1 moves. All moves were limited to using singular goals (e.g., moving closer to the pot, moving to get a coin, reaching a cloud to transport). Each team used conditionals once throughout the round.

Turn	Team 1	Team 2
Turn 1		
Codes:	• Move 1 step forward • **If on green, pick a clover card:** (Card: Stand next to the other team)	• Move 1 step forward • Move 2 steps forward • Move 3 steps forward • **If on a cloud, move to any other cloud**

Turn 2		
Codes:	• Move 1 step forward • Move 2 steps forward • Move 3 steps forward	• Move 4 steps forward • Rotate 90° • Move 2 steps forward • Rotate 90° • Move 3 steps forward
Turn 3		
Codes:	• Used the clover card (Card: "Stand next to the other team") • Move 4 steps forward	

In addition to the observable gameplay strategies, the students' engagement with sequencing and decomposition was also evident in their dialogues during the game. Table 3.2

provides a selection of transcript snippets that showcase these conversations. For instance, one excerpt captures a student breaking down the steps to reach a coin, while another illustrates a discussion about sequencing their moves for maximum efficiency. These conversations underscore the depth of the students' engagement with these programming concepts.

Table 3.2: Transcript excerpts illustrating student engagement in sequencing and decomposition.

Round / Team / Turn	Conversation	Description
	Examples of Sequencing	
Round 1 - Team 2 - Turn 1	● “So, I'm going to try to, I'm going to try to get to that cloud and then use this (card) to get all the way up here. Oh, okay. All right. So, move forward one step. Then two steps would be (here), ... and then three steps. Then on a, if on a cloud, then transfer to another cloud.”	A player is arranging actions in a specific sequence to achieve their goal of reaching a cloud.
Round 1 - Team 2 - Turn 2	● “We can go one step. Turn 90 degrees.” ● “Wait, wait. No, we turn 90 degrees here. One, two.”	The players are considering the order in which they should make moves and turns to navigate the board.
Round 2 - Team 2 - Turn 1	● “All right, so wait, one, two. If on a blue. So, we'd get a clover card and then 1, 2, 3 (steps), and then one, two. And then, uh, if on a blue, which we aren’t. 1, 2, 3 (steps).”	The player is thinking about the sequence of their moves, considering the conditions and steps they have available.
	Examples of Decomposition	

Round / Team / Turn	Conversation	Description
Round 1 - Team 1 - Move 2	● "So right now we're just going forward. Um, which would be good. 1, 2, 3, 1, 2, 1." ● "You'd be at 1, 1, 2, 1, 2, 3. You'd be at this (spot)."	The players break down their journey into smaller steps, ensuring that each step is clear and logical in order to reach the desired position on the board.
Round 2 - Team 2 - Move 1	● "Wait, wait, let's just see. Alright, so wait, one, two. If on a blue. So, we'd get a clover card and then 1, 2, 3, and then one, two. And then, uh, if on a blue, which we aren't. 1, 2, 3."	The player decomposes their main task into smaller subtasks, identifying the specific steps they need to take.
Round 2 - Team 1 - Move 1	● "So, I'm going to do move… move one step forward. This (spot). Move two steps forward? So that would be there. 1, 2, 1, 2, 3, and then I'm going to turn that way so I can do that, and then move two steps forward and then, [taps] 'go!' That's actually all I need to do."	The player breaks down their main goal into smaller tasks, deciding on individual moves and turns that will help achieve their goal.

Use of Conditionals

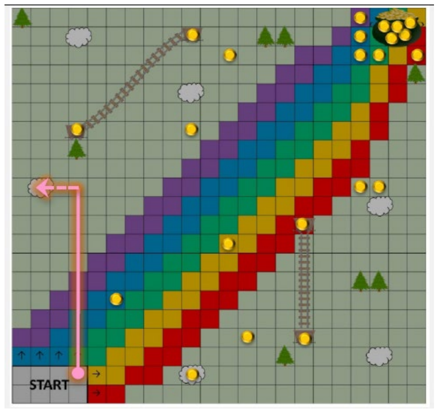
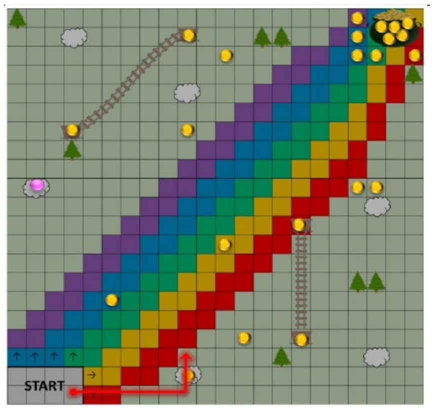
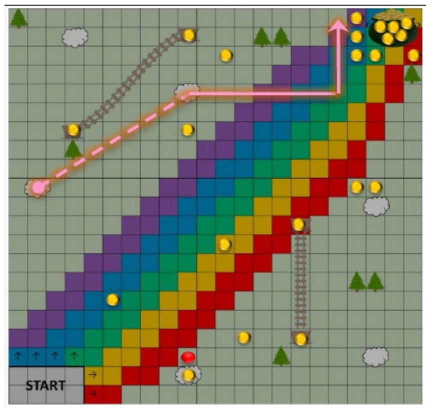
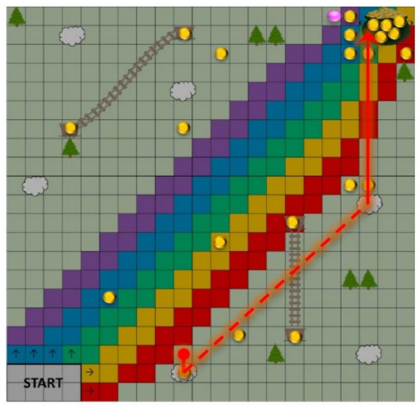
The game mechanics also facilitated participants' use of conditionals in their sequencing and decomposition strategies. In Round 1, as illustrated in Figure 3.5, Team 1 moved forward one square so they could use a color-conditional (i.e., "if on green…") to acquire a clover card (Team 1 - Turn 1). Team 2 followed by moving forward six squares so they could use a cloud-conditional ("if on a cloud, move to any other cloud") to move closer to the pot of coins at the far side of the game board (Team 2 - Turn 1). This strategy was followed by additional moves and rotates to move closer to the pot of coins (Team 2 - Turn 2). Nevertheless, Team 1 claimed victory in their third turn by utilizing their clover card to move next to their opponent and then proceed to the pot of coins (Team 1 - Turn 3).

In Round 2 participants exhibited an evolved use of programming skills. As depicted in Table 3.2 and Figure 3.6, teams began by decomposing the path to the pot of coins into smaller sub-goals. They then sequenced their tiles to achieve those sub-goals in a way that combined moving with higher-level programming concepts like conditionals. Team 2, for instance, started the round by using a ‘repeat’ function to move forward two squares, using a color-conditional to gain a clover card, and moving forward three squares twice. They then used the newly obtained Clover card to move to the nearest cloud (Team 2 - Turn 1).

In the second turn of Round 2, both teams constructed sequences of tiles that enabled longer movements across the board compared to the prior round. A notable aspect of their strategy was combining multiple movement tiles with the cloud-conditional. Utilizing the cloud-conditional in this way minimized the need for numerous 'move forward' tiles, making it a more resource-efficient strategy for moving around the board. Recognizing and leveraging this, Team 1 secured a win. They initiated by moving one square to a cloud, took advantage of the cloud-conditional to transition between clouds, and then moved forward nine squares, leading to them winning the round (Team 1 - Turn 2).

In the gameplay, students actively strategized to find optimal solutions, especially under the constraints of competition and time. They frequently considered alternative methods to achieve their goals, showcasing their adaptability and problem-solving capabilities.

Figure 3.6: Round 2 moves. Team 2 went first in the second round. There was increased use of conditionals to achieve multiple goals in one move. Conditionals were highlighted in bold.

	Team 2	Team 1
Turn 1		
Codes:	• **Repeat below 2 times** • Move 2 steps forward • **If on blue, pick a clover card:** (Card: "Move to the nearest cloud") • Move 3 steps forward • Rotate 90° • Use card: "Move to the nearest cloud"	• Move 1 step forward • **If on red, pick a clover card** • Move 2 steps forward • Move 3 steps forward • Rotate 90° • Move 2 steps forward
Turn 2		
Codes:	• Rotate 180° • Move 1 step forward • **If on a cloud, move to any other cloud** • Move 1 step forward • Move 3 steps forward • Move 2 steps forward • Move 2 steps forward • Rotate 90° • **If on green, pick a clover card** • Move 5 steps forward	• Rotate 180° • Move 1 step forward • **If on a cloud, move to any other cloud** • Rotate 180° • Move 5 steps forward • Move 2 steps forward • Move 2 steps forward

Learning to Do More Per Turn

Another programming skill that the game mechanics supported was learning to make use of a greater number of tiles per turn. The average number of tiles used per turn in the first round of gameplay was 3; of the tiles used, only two were conditionals (i.e., if-then). By the second round, the number of tiles used increased to an average of 6.75 per turn; this included five conditionals as well as the use of a loop (i.e., repeat). In terms of gameplay, the increase in tiles used per turn corresponded with each team establishing a greater number of goals per turn. For example, Team 1 was the only team to use a color-conditional to gain a clover card that ultimately won them the game in the first round. By the second round, both teams employed a color-conditional during their first turn as part of a longer sequence of moves (see Figure 3.6). Additionally, Team 2 embedded their color conditional in a repeat tile in order to gain a greater number of Clover Cards in a single turn. Team 2 also accomplished multiple goals in a single turn. They used a color-conditional in the first turn of the second round while also setting their position to use a cloud-conditional in their next turn. They were then able to win the game by combining the conditional with a sequence of moves that arrived at the pot of gold.

Discussion

The game mechanics of *Lucky Codes* allowed participants to practice essential programming concepts. Players were encouraged to actively engage in sequencing and decomposition, which are considered foundational in computer programming (Grover & Pea, 2018; Shute et al., 2017). As gameplay progressed, participants developed extended sequences of movements, breaking down the overarching task of reaching the pot of coins into smaller, manageable sub-goals. To achieve this, they visualized possible movements of the game piece in relation to available tiles, creating an executable sequence of codes. In this manner, the game piece became what Papert (1980) termed an "object-to-think-with" (p. 11), serving as a tangible representation that helped participants visualize how a computer program interacts with a programmable agent. This is particularly relevant for children's exploration of computer programming.

Drawing on external studies, Lister et al.'s (2004) multi-national research affirmed that novice programmers needed to learn to predict the outcomes of a computer program before advancing to more complex challenges. Other scholars (Bers et al., 2014; Blancas et al., 2020; Brennan & Resnick, 2012) echoed this, highlighting the benefits children derive from predicting the outcome of a programming sequence before diving into advanced programming concepts. Reflecting these findings, our game's dual-board system acts as a unique pedagogical design. One board is designated for programming, while the other board serves as a space where players can immediately enact the sequences they have coded on the first board. This two-board design mimics the experience of real-time feedback that digital platforms like Scratch integrate into the programming environment. Students can develop a section of code in the Coding Zone and then see the results play out in a tangible way on the game board. While there were instances where students made errors in enacting their codes, these mistakes were minimal due to the attentive oversight of other players, who were actively engaged in addressing any missteps.

Furthermore, our findings hint at the potential of a coding board game to introduce children to advanced programming concepts like conditionals. We intentionally designed our game so that our players did not need to piece together conditionals from multiple tiles; instead, they identified uses for a ready-made conditional statement. This design choice facilitated children's engagement with the concept of conditionals within the game framework. By the second round of gameplay, participants actively sought opportunities to incorporate conditionals into their strategy. This proactive approach likely stemmed from their realization of the benefits a conditional statement could offer in gaining a competitive edge. This finding is significant as it suggests that intentional game design can encourage gameplay extending beyond basic "move and rotate" mechanics found in many coding board games (Scirea & Valente, 2020; Wu, 2018). Focusing on using a concept like conditionals, rather than constructing a conditional statement, allowed us to engage participants in higher-level programming concepts in a meaningful and comprehensible way within the game context.

The way that our participants increased their use of tiles per turn from the first to the second round is also worthy of note. An important concept in computer programming is efficiency, meaning that the computer programmer does more with the limited time and space that is available within the computing system (Chowdhury et al., 2018; Grover & Pea, 2018). Our participants exhibited greater efficiency with their use of tiles in the second round. They repeatedly used conditionals to move farther or gain an advantage over the other team. They also found ways to combine their use of color-conditionals with other tiles to gain an advantage. Team 1 used a repeat function in the first turn of the second round to gain more clover cards. Likewise, Team 2 combined conditionals with sequences of moves that positioned them to win the game in the second round. These results suggest that *Lucky Codes* encouraged our participants to be more efficient with each turn - that is, it encouraged them to do more in a single round so that they could gain an advantage over their opponents. This finding is encouraging for proponents of board games to teach basic computing concepts to young children. Introducing algorithmic efficiency to children at an early age is important for supporting their ability to analyze and plan computer programming algorithms in the future (Kjällander et al., 2021).

Implications

For educators, this research suggests that incorporating board games like *Lucky Codes* into the classroom can be a valuable approach. It helps foster essential programming skills among students, including sequencing, decomposition, and conditionals, while also offering an enjoyable and interactive learning experience. This holds significant significance as the development of programming skills has become increasingly important in the modern-day workforce, where proficiency in computer programming, problem-solving, and critical thinking are highly sought after (Amri et al., 2019; SITRA, 2014). By integrating board games like *Lucky Codes*, educators can effectively equip their students with the necessary competencies to succeed in the 21st-century workforce (Sun et al., 2021; Yoon & Khambari, 2021).

For game designers, this study suggests it is possible to design games that teach programming skills while going beyond basic programming concepts like "move and rotate." Our game mechanics focused more on using a higher-level concept like conditionals than constructing the code associated with that concept. This may be one way to introduce children to higher-level programming concepts while reducing the abstract nature of those concepts. A shared game space may also help board games go beyond basic programming concepts. Our results suggest that each team learned the utility of conditionals by observing how the other team made use of them. In this way, our study helps address a noted gap in the literature about how specific components of educational board games relate to children's learning of programming skills through those games (Poole et al., 2022).

Limitations and Future Research

While our study offers valuable insights into the intersection of board games and programming concepts, it is essential to acknowledge its limitations. First, the research focused on a small group of four children, limiting the generalizability of findings to a broader population. Consequently, caution should be exercised when drawing conclusions about the wider demographic. Future studies with larger participant groups could yield a more comprehensive understanding of how *Lucky Codes* influences children's engagement with coding.

Furthermore, while our observations highlight the potential efficacy of board games in introducing children to programming concepts, the absence of explicit learning measures and pre-assessments restricts our ability to conclude on skill acquisition definitively. As an initial exploration, our gameplay duration was relatively brief, with our primary objective being to demonstrate how our game design exposed players to specific programming skills rather than conclusively proving skill improvement. Consequently, this paper lays the foundation for future investigations into the educational potential of the game. Subsequent research could delve into long-term changes in programming skills and attitudes.

Additionally, there is a gap in understanding the practical application of the skills acquired through games like *Lucky Codes* in real-world scenarios, especially during the transition to digital programming platforms like Scratch. Despite these limitations, our research is encouraging in that specific elements of our game design, such as incentivizing conditionals and using two game boards, resulted in a gameplay experience that introduced participants to core programming concepts in a manner exceeding many existing educational games. This study provides a springboard for future research exploring how the game facilitates the transition from board games to digital programming platforms and the enduring retention of skills gained, as compared to more traditional programming education methods.

Conclusion

Board games hold a unique potential to introduce children to foundational computer programming concepts. However, understanding the nuanced effects of specific game mechanics on children's learning remains essential (Poole et al., 2022). This study offers valuable insights into how specific game mechanics, like those in *Lucky Codes*, can encourage active engagement in essential programming practices. Through a detailed exploration of our game's design and its influence on children's application of programming skills, we seek to provide guidance to other game designers. Ultimately, we aspire to further establish board games as a potent tool for supporting elementary-level computer science education that is accessible to all children.

References

Amri, S., Budiyanto, C. W., & Yuana, R. A. (2019, December). Beyond computational thinking: Investigating CT roles in the 21st century skill efficacy. In *AIP Conference Proceedings* (Vol. 2194, No. 1, p. 020003). AIP Publishing LLC. https://doi.org/10.1063/1.5139735

Andersen, P. B., Bennedsen, J., Brandorff, S., Caspersen, M. E., & Mosegaard, J. (2003). Teaching programming to liberal arts students: a narrative media approach. *ACM SIGCSE Bulletin*, *35*(3), 109-113. https://doi.org/10.1145/961290.961543

Bandura, A., & Walters, R. H. (1977). *Social learning theory* (Vol. 1). Prentice Hall: Englewood cliffs.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? ACM Inroads, 2(1), 48-54.

Bell, T., & Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work?. In *Adventures Between Lower Bounds and Higher Altitudes (pp. 497-521)*. Springer, Cham. https://doi.org/10.1007/978-3-319-98355-4_29

Bell, T., Rosamond, F., & Casey, N. (2012). Computer science unplugged and related projects in math and computer science popularization. *The multivariate algorithmic revolution and beyond: Essays dedicated to Michael R. Fellows on the occasion of his 60th Birthday*, 398-456.

Berland, M., & Duncan, S. (2016). Computational thinking in the wild: Uncovering complex collaborative thinking through gameplay. *Educational Technology*, 29-35.

Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning*, 1(2), 65-81. doi:10.4018/ijgbl.2011040105. http://doi.org/10.4018/ijgbl.2011040105

Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. Journal of Computers in Education, 6(4), 499-528. https://doi.org/10.1007/s40692-019-00147-3

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145-157. https://doi.org/10.1016/j.compedu.2013.10.020

Bezemer, J. (2014). 14 Multimodal transcription: A case study. *Interactions, Images and Texts: A Reader in Multimodality. Berlin: de Gruyter Mouton*, 155-170. https://doi.org/10.1515/9781614511175

Blancas, M., Valero, C., Mura, A., Vouloutsi, V., & Verschure, P. F. (2020). "CREA": An inquiry-based methodology to teach robotics to children. In *Robotics in Education: Current Research and Innovations 10* (pp. 45-51). Springer International Publishing. https://doi.org/10.1007/978-3-030-26945-6_4

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). http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Cowan, K. (2014). Multimodal transcription of video: examining interaction in Early Years classrooms. *Classroom Discourse*, *5*(1), 6-21. https://doi.org/10.1080/19463014.2013.859846

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying Computational Thinking for Non–Computer Scientists (work in progress).

Ezeamuzie, N. O., & Leung, J. S. (2022). Computational thinking through an empirical lens: A systematic review of literature. *Journal of Educational Computing Research*, *60*(2), 481-511. https://doi.org/10.1177/07356331211033158

Georgia Department of Education. (2022). Georgia Standards of Excellence: Computer Science. https://www.georgiastandards.org/Georgia-Standards

Gibson, B., & Bell, T. (2013, November). Evaluation of games for teaching computer science. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 51-60). https://doi.org/10.1145/2532748.2532751

Gresse von Wangenheim, C., Araújo e Silva de Medeiros, G., Missfeldt Filho, R., Petri, G., Da Cruz Pinheiro, F., F. Ferreira, M. N., & Hauck, J. C. R. (2019). SplashCode - A Board Game for Learning an Understanding of Algorithms in Middle School. *Informatics in Education, 18*(2), 259-280. https://doi.org/10.15388/infedu.2019.12

Grover, S., & Basu, S. (2017, March). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 267-272).

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. Educational researcher, 42(1), 38-43.

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer Science Education*. https://doi.org/10.5040/9781350057142.ch-003

Guzdial, M. (2008). Education: Paving the way for computational thinking. Commun. ACM, 51(8), 25-27. https://doi.org/10.1145/1378704.1378713

Hajian, S. (2019). Transfer of learning and teaching: A review of transfer theories and effective instructional practices. *IAFOR Journal of education*, *7*(1), 93-111.

Harris, C. (2018). Computational Thinking Unplugged: Comparing the Impact on Confidence and Competence from Analog and Digital Resources in Computer Science Professional Development for Elementary Teachers. https://fisherpub.sjf.edu/education_etd/374

Hoffman, B., & Nadelson, L. (2010). Motivational engagement and video gaming: A mixed methods study. *Educational Technology Research and Development*, 58(3), 245-270. https://doi.org/10.1007/s11423-009-9134-9

Indiana Department of Education (2023). Indiana Academic Standards Computer Science: Kindergarten - Grade 8. https://www.in.gov/doe/students/indiana-academic-standards/science-and-computer-science/

Jiang, X., Harteveld, C., Yang, Y., Fung, A., Huang, X., & Chen, S. (2023). "If it's sunny, don't take an umbrella": a systematic evaluation of design principles for CT teaching games. *Educational Technology Research and Development*, 71, 1725–1763.

Kadar, R., Wahab, N. A., Othman, J., Shamsuddin, M., & Mahlan, S. B. (2021). A study of difficulties in teaching and learning programming: a systematic literature review. *International Journal of Academic Research in Progressive Education and Development*, *10*(3), 591-605. DOI:10.6007/IJARPED/v10-i3/11100

Kalmpourtzis, G. (2018). *Educational Game Design Fundamentals: A journey to creating intrinsically motivating learning experiences.* CRC Press.

Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons.

Kim, Y., Yoo, J., & Kim, N. (2019). *The Current Status and Improvement Plans for Elementary and Secondary Software Education.* National Assembly Research Service. https://www.assembly.go.kr/common/download.do?fid=bodo1&a.bbs_num=47966&file_num=43684&fpath=Bodo

Kjällander, S., Mannila, L., Åkerfeldt, A., & Heintz, F. (2021). Elementary students' first approach to computational thinking and programming. *Education Sciences*, 11(2), 80. https://doi.org/10.3390/educsci11020080

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking. Digital experiences in mathematics education, 3, 154-171.

Kuo, W. C., & Hsu, T. C. (2020). Learning computational thinking without a computer: How computational participation happens in a computational thinking board game. *The Asia-Pacific Education Researcher*, 29(1), 67-83. https://doi.org/10.1007/s40299-019-00479-9

Lee, V. R., Poole, F., Clarke-Midura, J., Recker, M., & Rasmussen, M. (2020, February). Introducing Coding through Tabletop Board Games and Their Digital Instantiations across Elementary Classrooms and School Libraries. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 787-793). https://doi.org/10.1145/3328778.3366917

Lindberg, R. S., Laine, T. H., & Haaranen, L. (2019). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games.

British Journal of Educational Technology, 50(4), 1979-1995. https://doi.org/10.1111/bjet.12685

Linneberg, M. S., & Korsgaard, S. (2019). Coding qualitative data: A synthesis guiding the novice. *Qualitative research journal*, *19*(3), 259-270. https://doi.org/10.1108/QRJ-12-2018-0012

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin, 36*(4), 119-150. https://doi.org/10.1145/1041624.1041673

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. https://doi.org/10.1016/j.chb.2014.09.012

Mayer, R. E. (2014). *Computer games for learning: An evidence-based approach*. MIT Press. https://doi.org/10.7551/mitpress/9427.001.0001

Mladenović, S., Krpan, D., & Mladenović, M. (2016). Using games to help novices embrace programming: From elementary to higher education. T*he International journal of engineering education*, 32(1), 521-531. https://www.researchgate.net/publication/291696663

Nambiar, R. (2020). Coding as an Essential Skill in the Twenty-First Century. In *Anticipating and Preparing for Emerging Skills and Jobs* (pp. 237-243). Springer, Singapore. https://doi.org/10.1007/978-981-15-7018-6_29

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: BasicBooks. http://www.medientheorie.com/doc/papert_mindstorms.pdf

Pecher, D., & Zwaan, R. A. (Eds.). (2005). *Grounding cognition: The role of perception and action in memory, language, and thinking*. Cambridge University Press.

Pierce, C.S. (1978), “Pragmatism and abduction”, in Hartshorne, C. and Weiss, P. (Eds), Collected Papers, Harvard University Press, Cambridge, MA, pp. 180-212.

Piteira, M., & Costa, C. (2013, July). Learning computer programming: study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication* (pp. 75-80). https://doi.org/10.1145/2503859.2503871

Plass, J. L., Homer, B. D., & Kinzer, C. K. (2015). Foundations of game-based learning. *Educational Psychologist,* 50(4), 258-283. https://doi.org/10.1080/00461520.2015.1122533

Plass, J. L., Mayer, R. E., & Homer, B. D. (Eds.). (2020). *Handbook of game-based learning*. MIT Press.

Poole, F. J., Clarke-Midura, J., Rasmussen, M., Shehzad, U., & Lee, V. R. (2022). Tabletop games designed to promote computational thinking. *Computer Science Education*, *32*(4), 449-475. https://doi.org/10.1080/08993408.2021.1947642

Rahmat, M., Shahrani, S., Latih, R., Yatim, N. F. M., Zainal, N. F. A., & Ab Rahman, R. (2012). Major problems in basic programming that influence student performance. *Procedia-Social and Behavioral Sciences*, *59*, 287-296. https://doi.org/10.1016/j.sbspro.2012.09.277

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67. https://doi.org/10.1145/1592761.1592779

Rijke, W. J., Bollen, L., Eysink, T. H. S., & Tolboom, J. L. J.(2018). Computational Thinking in Primary School: An Examination of Abstraction and Decomposition in Different Age Groups. Informatics in Education, 17(1), 77-92. doi:10.15388/infedu.2018.05

Scirea, M., & Valente, A. (2020, September). Boardgames and computational thinking: how to identify games with potential to support CT in the classroom. In *Proceedings of the 15th International Conference on the Foundations of Digital Games* (pp. 1-8). https://doi.org/10.1145/3402942.3409616

Selby, C. C. (2012, November). Promoting computational thinking with programming. In *Proceedings of the 7th workshop in primary and secondary computing education* (pp. 74-77). https://doi.org/10.1145/2481449.2481466

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, *22*(2), 469-495. https://doi.org/10.1007/s10639-016-9482-0

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

SITRA (2014). *Future will be built by those who know how to code*. https://www.sitra.fi/en/articles/future-will-be-built-those-who-know-how-code/

Sun, D., Ouyang, F., Li, Y., & Zhu, C. (2021). Comparing learners' knowledge, behaviors, and attitudes between two instructional modes of computer programming in secondary education. *International journal of STEM education*, *8*(1), 1-15.

Sun, L., Guo, Z., & Hu, L. (2021). Educational games promote the development of students' computational thinking: a meta-analytic review. *Interactive Learning Environments*, 1-15. https://doi.org/10.1080/10494820.2021.1931891

Taub, R., Armoni, M., & Ben-Ari, M. (2012). Cs unplugged and middle-school students' views, attitudes, and intentions regarding cs. *Acm Transactions on Computing Education*, 12(2), 1-29. https://doi.org/10.1145/2160547.2160551

Tsarava, K., Moeller, K., & Ninaus, M. (2018). Training computational thinking through board games: The case of Crabs & Turtles. *International Journal of Serious Games*, *5*(2), 25-44. https://doi.org/10.17083/ijsg.v5i2.248

Vihavainen, A., Paksula, M., & Luukkainen, M. (2011, March). Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 93-98). https://doi.org/10.1145/1953163.1953196

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

Wong, G. K., Cheung, H. Y., Ching, E. C., & Huen, J. M. (2015, December). School perceptions of coding education in K-12: A large scale quantitative study to inform innovative practices. In *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 5-10). IEEE. https://doi.org/10.1109/TALE.2015.7386007/

Wu, L., Looi, C. K., Multisilta, J., How, M. L., Choi, H., Hsu, T. C., & Tuomi, P. (2020). Teacher's perceptions and readiness to teach coding skills: a comparative study between Finland, Mainland China, Singapore, Taiwan, and South Korea. *The Asia-Pacific*

Education Researcher, *29*(1), 21-34. https://doi.org/10.1007/s40299-019-00485-x

Wu, S. Y. (2018, August). The development and challenges of computational thinking board games. In *2018 1st international cognitive cities conference (IC3)* (pp. 129-131). IEEE. Doi: 10.1109/IC3.2018.00-45.

Yang, D., & Kopcha, T. J. (2022). Designing a Board Game for Beginning Block-Based Programmers. *International Journal of Designs for Learning*, *13*(1), 35-45. https://doi.org/10.14434/ijdl.v13i1.32211

Yoon, C. S., & Khambari, M. N. M. (2021) Robobug: An unplugged approach to computational thinking. *Proceedings of the International University Carnival on e-Learning (IUCEL) 2021*. 333-335.

CHAPTER 4

HOW DO BOYS AND GIRLS LEARN COMPUTATIONAL THINKING THROUGH A BOARD GAME? A CASE STUDY OF *LUCKY CODES* [3]

[3] Yang, D. & Kopcha, T. To be submitted to *Educational Technology Research and Development* or *International Journal of Game-Based Learning*

Abstract

This study investigates the use of the educational board game *Lucky Codes* to teach computational thinking skills to boys and girls in grades four to seven, addressing the gap in understanding how unplugged games influence learning. The research aimed to explore (1) general gameplay trends, (2) changes in computational thinking knowledge over time, and (3) differences in attitudes and engagement by gender. An exploratory case study design was implemented using a mixed methods approach, engaging 17 students in three gameplay sessions. Quantitative data were collected through pre- and posttests and gameplay analysis, while qualitative data included observations, surveys, and interviews. Results indicated that students increasingly used advanced coding concepts such as loops and conditionals over time, suggesting improvements in their use of computational thinking skills. Both boys and girls preferred the strategic elements of the game, but boys were more drawn to the competitive aspects, while girls placed greater emphasis on social interactions and collaborative dynamics. The study concludes that unplugged games like *Lucky Codes* may effectively support learning, with the potential for even greater impact through adaptive challenges and reflective activities. While the small sample size was a primary limitation of the study, future research with more diverse participants could provide deeper insights into its broader applicability.

Introduction

Over the last twenty years, computer science (CS) has evolved from a niche subject for future engineers and programmers to an essential part of the K-12 curriculum for learning computational thinking (Grover & Pea, 2013; Wing, 2006). Computational thinking (CT) is a process that involves solving problems, designing systems, and understanding human behavior through the lens of computer science principles (Wing, 2006). One way that CT has been taught to children is through computer programming. The act of learning to program introduces students to CT skills such as algorithmic thinking, debugging, and sequencing (Lye & Koh, 2014; Resnick et al., 2009; Shute et al., 2017).

Educational games have become known as an effective means of teaching programming concepts. Using games for learning, in general, can increase students' interest, which often leads to improved performance (Abrantes et al., 2007; Harackiewicz et al., 2016; Lathifah et al., 2023; Tsarava et al., 2018). Unplugged games, which are hands-on activities that do not use computers, are especially beneficial for younger students as they can help build a more concrete understanding of abstract CT concepts using tangible materials (Barsalou & Wiemer-Hastings, 2005). Multiple studies have measured the learning effects of various unplugged activities and found that they can enhance CT performance in K-12 classrooms (Chen et al., 2023; Chongo et al., 2021; Merino-Armero et al., 2022; Polat & Yilmaz, 2022; Zhang et al., 2024).

However, there are still limitations to understanding how the learning happens as many studies predominantly rely on pre- and posttests, offering limited insight into how students' gameplay evolves and changes across multiple sessions (Chen et al., 2023; Tang et al., 2020; Zhang & Nouri, 2019). Pre- and posttests are often limited in their ability to measure the deeper, evolving development of students' understanding and application of CT skills, as they tend to focus on specific outcomes rather than the complex and nuanced processes of learning (Black & Wiliam, 1998; Hattie & Timperley, 2007). These assessments could miss the dynamic process of learning that occurs during gameplay, such as the development of problem-solving strategies, iterative thinking, and collaborative skills, which are crucial components of CT (Grover & Pea, 2013; Shute et al., 2017). Moreover, using only pre- and posttests could fail to provide insights into how students adapt their thinking in response to challenges encountered during gameplay or how their engagement and motivation influence their thought developments across multiple sessions (Kafai & Burke, 2015).

Additionally, it is important to consider gender differences in game preferences, which can influence engagement and learning outcomes. Studies have found boys and girls often favor different types of games: boys typically preferred faster-paced action and strategic games, whereas girls were more inclined toward creative and social games (Aleksić & Ivanović, 2017; Hartmann & Klimmt, 2006; Kinzie & Joseph, 2008; Lange et al., 2021; Nguyen et al., 2023). However, it is unclear whether these preferences apply to unplugged gaming contexts and how each gender reacts to a game designed to be gender-neutral (e.g. games that include features such as using animal figurines instead of gendered characters and incorporate game dynamics that do not overly emphasize action, strategy, creativity, or social interaction).

This study seeks to address these gaps by examining the engagement and learning experiences of boys and girls with *Lucky Codes*, a gender-neutral educational board game developed by the authors. The game encourages players to build codes using CT skills such as sequencing, loops, conditionals, and debugging. Using this game as the intervention, this study aims to explore how gameplay trends evolve over multiple sessions and how gender preferences influence engagement and learning outcomes. The findings will provide a deeper understanding of the dynamics and effectiveness of unplugged games in developing CT skills.

The research questions guiding this study are: (1) What are the general trends in gameplay among students using *Lucky Codes*? (2) How does students' CT knowledge evolve over time, and how do these changes vary by gender? (3) What are the patterns in students' attitudes during the gameplay of *Lucky Codes*, and how do these differ by gender?

Literature Review

Programming Education and Computational Thinking

Computer science (CS) education in K-12 classrooms has seen notable changes due to the increasing role of technology in our daily lives. What was once a specialized field for aspiring engineers and programmers has now become a crucial skill for all students to navigate and interact with today's digital world (Grover & Pea, 2013; Wing, 2006). A key part of this shift is the focus on computational thinking (CT), a concept highlighted by Jeanette Wing in the mid-2000s. For Wing (2006), CT is about “thinking like a computer scientist.” (p. 34), which goes beyond the ability to program a computer. Wing defined CT as "solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science" (p. 33). This involves skills such as thinking abstractly, decomposing problems, working with algorithms, and effectively representing and solving computational problems using a computing machine.

Since then, scholars have refined the definition and components of CT. While the details vary, common components include abstraction, sequence, conditional, algorithm, loop, parallelism, and debugging (Ezeamuzie & Leung, 2022). These skills are valuable not only for CS careers but also for improving problem-solving, critical thinking, and analytical abilities across disciplines (Wing, 2006). The growing emphasis on CT has prompted numerous scholars to advocate for its integration into K-12 education (Barr & Stephenson, 2011; Grover & Pea, 2013; Lye & Koh, 2014). This call to action has gained global recognition, with countries like the United States, England, Australia, France, Israel, and South Korea integrating CT into their core curriculum (Lindberg et al., 2019; Nambiar, 2020; Wu et al., 2020).

Incorporating programming education within CS classes serves as a powerful tool for introducing students to CT concepts, as it introduces students to CT by developing problem-solving skills, engaging them in hands-on applications of CT concepts like sequencing, conditional thinking, and debugging, and fostering systematic, transferable thinking (Lye & Koh, 2014; Resnick et al., 2009; Selby, 2012; Shute et al., 2017; Wing, 2006). Block-based programming, such as Scratch, is known to be particularly beneficial because it simplifies the coding process, making it more accessible and engaging, especially for novice learners (Resnick et al., 2009). Multiple empirical studies advocate for these benefits. For example, Moreno-León et al. (2016) discovered that using Scratch coding led to substantial improvements in CT skills across multiple subjects. Similarly, Aksit and Wiebe (2019) and Bilgic and Dogusoy (2023) both found that block-based programming (such as Scratch) has positively impacted students' CT scores. Some scholars, like Moors et al. (2018), have addressed concerns about block-based programming, noting that it can lead to reduced confidence when transitioning to text-based languages and may cause misconceptions due to reliance on visual cues. However, systematic reviews, such as by Zhang and Nouri (2019) and Jin and Cutumisu (2024), indicate that block-based programming remains a highly popular and effective means of CT learning in K-12 education.

Unplugged Games for Programming

As the emphasis on CT has grown in K-12 education, the need for effective and accessible learning methods has become increasingly important. Unplugged activities, such as board games, have emerged as a powerful tool for CT education. Chen et al. (2023), in their systematic review of 49 studies, found that unplugged activities significantly improve CT skills by providing hands-on, interactive learning experiences that make abstract concepts more concrete. Similar studies have found that the physical game pieces used in unplugged activities provide a tangible way for students to interact with abstract CT concepts, fostering more profound understanding and long-term retention (Barsalou & Wiemer-Hastings, 2005). These visual aids help students bridge the gap between abstract ideas and real-world applications, making the learning experience more meaningful and relevant (Monga et al., 2018; Turchi et al., 2019).

The hands-on and tangible nature of unplugged activities also fosters a collaborative learning environment where all students can actively participate. They can observe each other's strategies, share insights, and work together to solve problems. This contrasts with digital learning environments, which often limit collaboration as only one student typically controls the mouse or keyboard at a time. The collaborative atmosphere in unplugged activities not only enhances focused behavior and teamwork but also plays a crucial role in developing communication and interpersonal skills, which are essential for both academic success and real-world applications (Küçükkara & Aksüt, 2021; Marjanen et al., 2011; Rist et al., 2006).

Finally, the accessibility of unplugged activities is a critical advantage, particularly in diverse educational settings. By removing the need for technology, unplugged activities ensure that all students, regardless of their access to digital devices, can engage with and develop essential CT skills (Bell & Vahrenhold, 2018; Gibson & Bell, 2013). This inclusivity is crucial for bridging educational gaps, especially in under-resourced schools where access to technology may be limited. By making CT education available to a broader range of students, unplugged activities play a vital role in promoting equity in education (Sun et al., 2021).

Research on CT Board Games

Multiple scholars have developed CT board games for K-12 students and studied their effects on learning. For instance, Kuo and Hsu's (2020) *Robot City* was designed to help students apply CT skills, develop problem-solving strategies, and collaborate on constructional tasks. Their research found that students' learning outcomes were better when they engaged in clear-ended collaborative tasks rather than open-ended competitive tasks. Similarly, Tsarava et al.'s (2018) *Crabs & Turtles* and Yoon and Khambari's (2022) *Robobug* were created to offer an accessible introduction to CT. These studies, which involved adult players and experts in testing the games, revealed their positive potential for fostering essential computational thinking skills, such as logical reasoning, algorithmic thinking, and problem-solving in children. Gresse von Wangenheim et al.'s (2019) *SplashCode* is a low-cost board game designed to reinforce basic algorithms and programming concepts. Their findings showed that the game significantly improved students' understanding of algorithms, enhanced their motivation and engagement, and promoted positive social interactions, making the learning process both effective and enjoyable.

While studies on board games generally indicate that they can effectively teach CT skills, they often lack depth in exploring how these games influence students' learning processes. Most existing research predominantly relies on pre- and posttests and observational data, with few studies employing qualitative methods like interviews to delve into participants' in-depth thinking processes (Chen et al., 2023; Tang et al., 2020). Qualitative approaches are crucial for gaining a more nuanced and accurate understanding of how learning is experienced by players, especially when the participants are children. As Flewitt (2013) points out, relying solely on observational data can introduce researcher biases in interpreting participants' behaviors, whereas interviews allow participants to articulate their actions and perspectives, uncovering complexities and differences that might otherwise be overlooked. Understanding those complexities and differences is crucial for advancing CS education. As emphasized by scholars, there is a need for studies that not only determine which aspects of CT can be effectively taught through educational board games but also explore how specific game elements can be designed to enhance the development of CT skills during gameplay (Brennan & Resnick, 2012; Poole et al., 2022; Videnovik et al., 2023). Such in-depth research can inform the creation of more effective educational tools that align with the nuanced learning processes of students, ultimately advancing both practice and research in CS education.

Gender and Games

Gender disparities in CS education reveal both critical challenges and opportunities for promoting CT among all students. The gender gap in education is particularly pronounced in CS compared to other subjects like mathematics, biology, and chemistry (Jaccheri et al., 2020; Master et al., 2021). Studies have found that boys tend to show higher self-efficacy and confidence in programming-related subjects compared to girls, which leads to greater participation in related activities (Kallia & Sentance, 2018; Tuğtekin et al., 2018; Zdawczyk & Varma, 2022). Lower self-efficacy in girls can negatively impact their motivation, learning success, and interest in future engagement in CS courses (Srisupawong et al., 2018). Therefore, it is important to design educational experiences that intentionally promote interest and confidence in learning CT for both girls and boys, ensuring that all students have equal opportunities to succeed (Master et al., 2021).

To design effective educational interventions, it is essential to recognize the different ways boys and girls engage with CT games. Given that girls generally express less interest in the subject compared to boys, multiple studies have shown that intentionally designed learning interventions can successfully promote girls' interest and confidence in learning CT (Barker & Aspray, 2006; Kuo & Kuo, 2023; Ma et al., 2021). Achieving this in the context of CT games requires a thorough understanding of how girls interact with the game and their preferences to ensure it appeals to both genders.

Research on gaming preferences in digital games provides insights that could inform the design of unplugged games. Research findings indicate that while both genders enjoy problem-solving games (Kinzie & Joseph, 2008), boys generally prefer action-oriented and fast-paced games, such as sports, first-person shooters, and fighting games, along with active strategy and strategic play modes (Chang & Chen, 2023; Hamlen, 2011; Homer et al., 2012; Kinzie & Joseph, 2008; Nguyen et al., 2023). In contrast, girls tend to favor simulations, educational games, puzzles, social interaction elements, and creative and exploratory games (Chang & Chen, 2023; Hamlen, 2011; Homer et al., 2012; Kinzie & Joseph, 2008; Nguyen et al., 2023).

While these findings provide valuable insights into boys' and girls' gaming preferences, there is little understanding regarding whether these preferences and engagement patterns apply to unplugged gaming contexts such as board games. Unplugged games, which involve physical interaction, face-to-face communication, and hands-on activities, offer unique engagement dynamics that differ from digital games (Bell & Vahrenhold, 2018). Understanding how boys and girls interact with unplugged games and whether the same gender-specific preferences are observed is crucial for designing effective educational tools in non-digital settings. This gap highlights the need for further research to explore the applicability of these findings in unplugged gaming contexts.

Method

Study Design

This study employed an exploratory case study design. According to Creswell and Plano Clark (2017), a case study involves an in-depth investigation of a single unit or system within its real-life context, making it particularly beneficial for gaining a holistic understanding of specific phenomena. Yin (2018) reinforces the value of case studies in exploring contemporary phenomena where the boundaries between the phenomena and their context are not clearly evident. This design is especially useful when existing research on the phenomenon is limited, as it allows researchers to investigate the phenomenon from multiple perspectives using various data sources, such as interviews, observations, and documents (Creswell & Plano Clark, 2017). The goal of an exploratory case study is not to test a specific hypothesis but to explore the phenomenon in depth and generate new insights or theories (Yin, 2018). This approach is particularly beneficial in the early stages of research when the aim is to identify key issues in depth, generate hypotheses, or develop a theoretical framework (Yin, 2018). The case explored in this study was the educational board game *Lucky Codes*.

Data were collected and analyzed using an embedded mixed methods approach, which integrates both quantitative and qualitative methods within the case study design to allow each to complement the other (Creswell & Plano Clark, 2017). According to Creswell and Plano Clark (2017), embedded mixed methods involve incorporating one type of data within a larger framework driven by another primary method, enabling researchers to address complex research questions from multiple perspectives. In this study, quantitative data, such as pre- and post-test assessments and gameplay metrics, provided measurable insights into changes in students' computational thinking skills over time, helping to identify *what* areas required closer examination in the qualitative analyses. Meanwhile, qualitative data, including detailed gameplay observations and open-ended survey responses, captured the nuanced ways students engaged with the game and the reasoning behind their coding choices, offering the *why* behind the numbers from the quantitative analysis. A visual representation of this design is shown in Figure 4.1.

By embedding these qualitative insights within the quantitative framework, the study was able to cross-validate findings and explore the underlying reasons behind observed trends (Creswell & Plano Clark, 2017). Blending quantitative and qualitative data served to triangulate the findings between objective and subjective data sources, which is recognized as a way to increase the credibility of the results and the trustworthiness of the analysis (Creswell & Creswell, 2017).

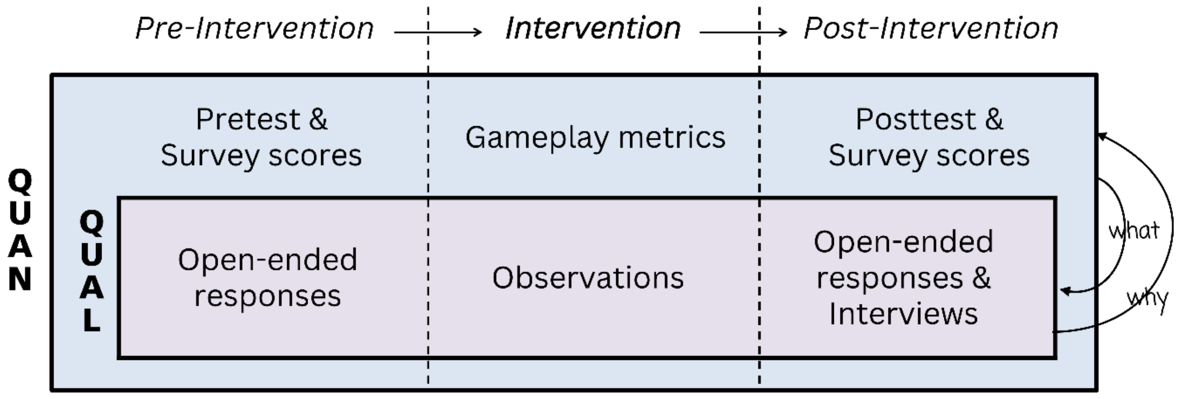


Figure 4.1: The embedded mixed methods study design of this study.

The Game, *Lucky Codes*

The intervention used for this study was called *Lucky Codes*, an educational board game designed to teach coding concepts to novice students, specifically targeting upper elementary school students. The game was developed through several rounds of prototyping, which is detailed in Yang and Kopcha (2022). The main purpose of the game's development was to offer teachers and students a fun and gender-neutral game that encourages players to engage in higher-level coding concepts, such as loops and conditionals, in addition to fundamental concepts like sequencing and debugging. In the game, students compete against one another in teams to acquire gold coins scattered about a game board. The winning condition is to collect five coins before the other team.

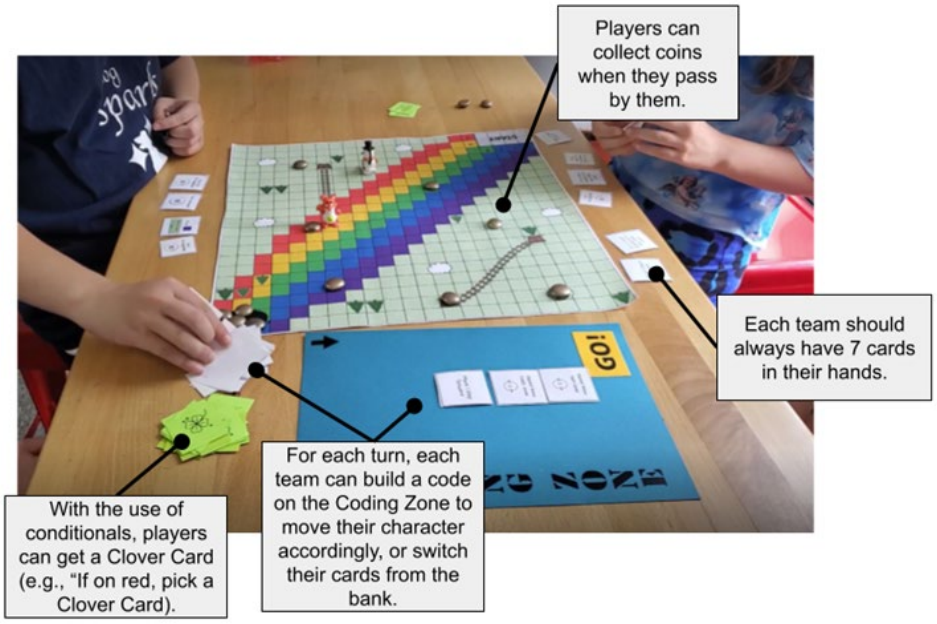

Players can collect coins when they pass by them.
Each team should always have 7 cards in their hands.
For each turn, each team can build a code on the Coding Zone to move their character accordingly, or switch their cards from the bank.
With the use of conditionals, players can get a Clover Card (e.g., "If on red, pick a Clover Card).
GO!

Figure 4.2: Screenshot of students playing *Lucky Codes* with descriptions of gameplay. The game consists of two game boards: Action Board (above) and Coding Zone (below).

As displayed in Figure 4.2, there are two game boards: Coding Zone and Action Board. Players create a sequence of tiles on the Coding Zone that their game pieces (animal figurines) will execute on the Action Board. The two-board dynamic mirrors the way coding takes place in block-based programming environments such as Scratch. The tiles used in the Coding Zone include common coding commands, such as movements ("turn," "move"), conditionals ("if …, then, …"), and loops ("repeat"). Players can use two types of conditionals in the game: cloud-conditionals (i.e., "If on a cloud, transport to another cloud"), which allow players to jump around clouds across the Action Board, and color-conditionals (i.e., "If on [a specified color], pick a Clover Card"), which are necessary to obtain "Clover Cards." Clover Cards are random command cards (e.g., "Move two steps diagonally in any direction," "Exchange one code tile with the other team.") that add an element of chance to the gameplay. These cards are desirable because they can be used at any time by the players like a wild card. Clover Cards can only be obtained when using conditionals, thereby incentivizing players to engage with more advanced coding concepts beyond basic pathfinding movements.

A typical game turn begins with a team either creating a sequence using the code tiles in their hands to move their character to a desired spot or exchanging any unneeded tiles for new ones from the draw pile (also referred to as the “bank”). When a team chooses to create a code sequence on the Coding Zone, the players move their game piece on the Action Board according to the sequence they created. When players pass by or land on coins on the Action Board, they collect them, which serve as their points in the game. Once the code is executed, the opposing team gets a turn. The gameplay repeats until a team collects five coins, either by going around the game board or reaching the pot of gold at the end of the rainbow that is initially filled with five coins (players can transfer some gold coins from the pot to different locations during gameplay using certain Clover Cards).

Participants

An initial group of 23 students from grades four to seven was identified as potential participants in the study. These students attended a private STEAM (Science, Technology, Engineering, Arts, and Mathematics) school in Georgia, which had a total enrollment of 51 students. The school curriculum regularly engaged students in hands-on STEM explorations, including basic CT concepts. However, no dedicated computer programming or coding classes were offered.

The school used the IOWA Assessments to measure academic progress, which is a norm-referenced test that compares a student's performance to a nationally representative sample of their grade level, with scores between the 25th and 75th percentiles considered average (Riverside Insights, n.d.). At the time of this study, the participants' scores in English/Language Arts (ELA) and Math indicated that most students performed at or slightly above the national average. Specifically, the average scores for grades 4, 5, 6, and 7 were as follows: Grade 4 (ELA = 76%, Math = 79%), Grade 5 (ELA = 78%, Math = 53%), Grade 6 (ELA = 75%, Math = 60%), and Grade 7 (ELA = 75%, Math = 75%).

The final sample consisted of 17 students who met three key criteria: parental consent was granted, the student was willing to participate, and the student provided verbal assent to participate in the study. Most students (70%) had previous introductory exposure to coding through school clubs or one-time activities, mainly using Scratch, with some also having limited experience in platforms like Code.org, Lego Robotics, or Python.

The recruitment process began with researchers explaining the study's goals and procedures. Interested students were given parental permission forms detailing the study, emphasizing voluntary participation and the ability to withdraw from the study at any time without any consequences. In the subsequent week, only the students who returned signed forms and agreed to participate were included in the study. This resulted in a group of 10 male and 7 female students. All study procedures were approved by the University of Georgia's Institutional Review Board on April 28, 2023.

Procedures and Data Collection

The game sessions for data collection took place over three days; these sessions were held during students' regular class time. Figure 4.3 shows the timeline of the data collection procedures. The first day consisted of a pretest, the first round of gameplay, and then an interest survey that examined students' interests in playing the game and learning coding. Day 2 consisted of another round of gameplay and an interest survey. Day 3 consisted of the final round of gameplay, a final interest survey with a posttest, and interviews with selected participants. All gameplay sessions were video recorded with prior agreement from the participating students.

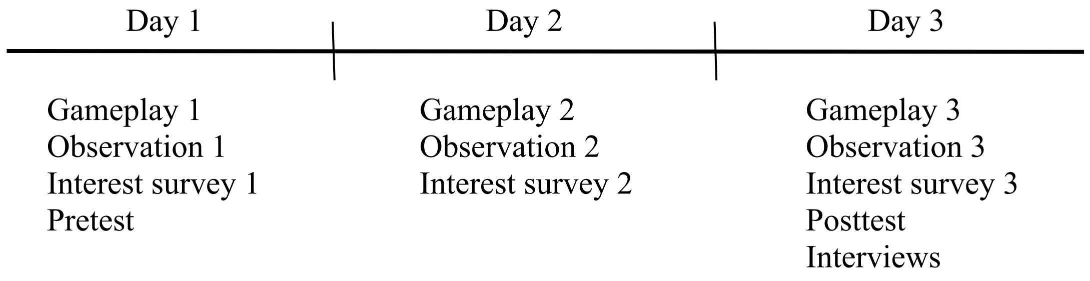


Figure 4.3: Data collection timeline.

Students were organized into groups of three or four, with each group divided into two teams. Groups were formed by taking into account the students' proximate grade levels to ensure balanced gameplay. This grouping strategy acknowledges that while cognitive development can vary among individuals, research suggests that cognitive skills, which are crucial for academic achievement, generally increase with age (Boman, 2023; Cleveland et al., 2022). Therefore, given the general correlation between age and cognitive development, grade-based grouping was deemed the most practical and effective approach for this study.

Within each group, students were allowed to choose their own partners and had the option to switch partners between rounds. This decision was informed by studies suggesting that allowing students to select their partners promotes comfort and enhances their performance potential (Choi, 2015; Hartl et al., 2015; Zhong et al., 2016). When grouping students, the balance of gender within a group was not the highest priority than the grade level and students' preference of partners. This approach was supported by Admiraal et al. (2014), who found that gender composition had no significant effect on learning experiences or outcomes. Additionally, the practical consideration of an unbalanced number of boys and girls participating in this study further justified the approach.

Since the game sessions took place during regular class hours, some students in the classroom who were not participating in the study were included in the groups to maintain an inclusive environment. This arrangement ensured that all students had the opportunity to be a part of the gameplay sessions unless they personally chose not to participate. However, in groups with a mix of participating and non-participating students, their gameplay was not video-recorded to respect the privacy of the non-participating students. Only the survey, tests, and interview data were collected from the participating students in those groups. As a result, video-recorded observation data was collected from four groups with the following gender distribution: Groups 1 and 3 included three males, and Groups 2 and 4 had three females and one male. Two male and one female participant were in groups with non-participating students and were therefore not video recorded. The following sections detail each type of data collected.

Gameplay Observation

During the gameplay sessions, students' gameplay actions and conversations were video-recorded during the gameplay sessions using tablets and smartphones on tripods. The recordings captured the codes that students constructed on the Coding Zone, the corresponding movements on the Action Board, and students' verbal discussions.

Pre- and Posttest

The participants completed a pre- and posttest of their programming knowledge (see Appendix B). The pre-test was administered on the first day before playing the game and the posttest was administered on the third day after playing the last game. All students were given 20 minutes to complete each of the tests. Before completing the tests, students were informed about the purpose of the tests and assured that their results would be used only for research purposes.

The pre- and posttests consisted of five items developed by the researcher, designed to evaluate students' understanding of the programming skills addressed through gameplay. These items were based on the game's main learning goals of the game: to read and create functional code sequences, to use loops and conditionals to enhance programming efficiency, and to identify and correct errors in codes.

To enhance the relevance and familiarity of the test for the students, each item incorporated images from the game boards. These images helped situate the questions within the gameplay context, making the test items more intuitive and relatable for the students. A sprite image from Scratch was borrowed to represent the game piece in the test scenarios. The tasks for each test item were as follows:

1. Sequencing: Assembling movement codes in a specific order.
2. Conditionals and Loops: Interpreting a series of codes that incorporate conditionals within a loop.
3. Loop Efficiency: Employing loops in assembling codes to achieve a specified goal efficiently.
4. Advanced Efficiency: Resolving a more complex version of the problem presented in the previous item.
5. Debugging: Identifying and correcting an error in a pre-written code.

The tests were scored using a rubric (Appendix C), with each item assigned up to 4 points, for a total possible score of 20. To illustrate how a test item was scored, test item 3 required students to collect five coins using the smallest number of code blocks possible. The ideal solution (4 points) involved using a loop to repeat the actions of moving forward, turning, moving forward, collecting the coin, and turning, five times. However, some students managed to collect all five coins by manually repeating the movements multiple times without using loops. These responses were scored 3 points because, while the task was successfully completed, the solution lacked efficiency and could have been improved by utilizing loops. Students who omitted a part of the sequence, such as leaving out the final turning, received 2 points, as they demonstrated progress and partial understanding, but the errors prevented the students from fully achieving the goal.

Survey

Students completed an interest survey at the end of each game session to assess (1) the level of their enjoyment in playing the game and (2) their interest in learning coding (see Appendix D). Enjoyment level was measured by a 7-point visual analog scale with the question, “How much did you enjoy the game so far?” The scale consisted of a horizontal line with face emojis representing varying levels of enjoyment. The scale was anchored at one end with a frowning face to represent "Not Enjoyable at All" (scored as 1) and at the other end with a smiling face to represent "Extremely Enjoyable" (scored as 7). Participants were instructed to mark one of the faces on the line that best represented their level of enjoyment and interest. The survey also included two open-ended questions to allow participants to explain what they found the most and least enjoyable aspects of the game.

Then, participants completed a five-item scale to measure students’ interest in coding after each game session. The scale was drawn from the Elementary Student Coding Attitudes Survey, which had previously been established and validated by Mason and Rich (2020). Specifically, the questions used in this study were from the “Coding Interest” category of their survey. Participants answered statements such as, “I would like to learn more about coding” or

"Solving coding problems seems fun," indicating their level of interest on a five-point Likert scale that ranged from strongly disagree (1) disagree (2), somewhat agree (3), agree (4), to strongly agree (5). All five items were added to create a total interest in learning coding, where a higher value indicated a high level of interest. In the current study, this scale exhibited an acceptable level of internal consistency (*a* = .718) (Nunnally & Bernstein, 1994).

Interviews

On the last day, semi-structured interviews were conducted with 12 students. Priority was given to students who demonstrated a noticeably high or low interest in the game or those whose survey responses could use additional explanations. These interviews served as complementary data to gain deeper insights into students' gameplay experiences and responses to the interest surveys. The conversations specifically addressed students' reactions to the challenges of the coding tasks, their experiences with various elements of the game, and their overall enjoyment and frustration with coding activities.

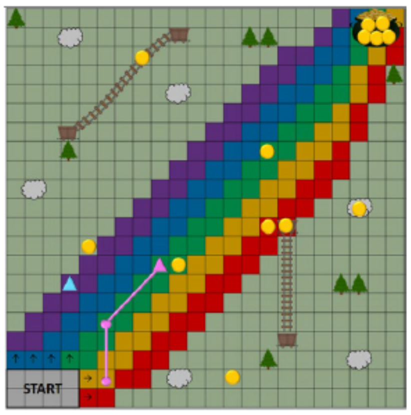
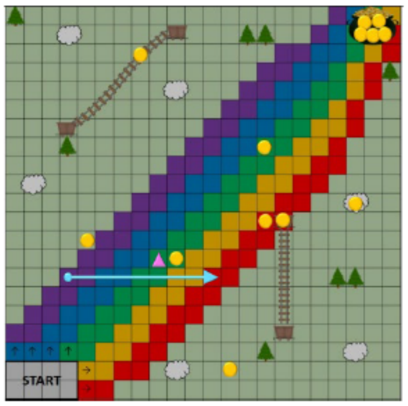
Data Analysis

Gameplay Observation

The video recordings of the participants' gameplay were first transcribed in a document that included written records of the codes students used each turn, the corresponding visual representations of the gameplay, and transcripts of students' conversations for each turn. This format provided a holistic view of students' interaction with the game over time. Figure 4.4 presents an example of a part of the transcript; the first team to move was called Team A (tracked with pink arrows), and the second was called Team B (tracked with light blue arrows). The triangular arrowheads indicate the direction in which each team's game piece is facing during each turn. Note that some moves may appear disconnected between turns, as game pieces could sometimes be moved by the opposing team's use of Clover Cards. This method of visually representing the gameplay movements was chosen to enhance the observational data by offering a clear visual trajectory of each team's progress throughout the game. As Zallio (2021) suggests, such visual representation improves the clarity and impact of the presentation, making the information more accessible and engaging to a broader interdisciplinary audience.

Using this transcript, students' coding behaviors were analyzed and grouped into four emergent code patterns through a content analysis of their gameplay interactions. Content analysis is a systematic approach to examining communication artifacts (in this case, coding sequences used during gameplay) in which codes are assigned and categorized to help interpret meaningful data (Krippendorff, 2013). As a result, four distinct coding patterns were identified: 1) Movements Only, involving only directional commands; 2) Conditional with Movements, which integrates conditions with movement commands; 3) Loops with Movements, combining loops with spatial navigation; and 4) Loops and Conditional with Movements, which merges loops, conditions, and movements.

Figure 4.4: Example of the visual transcript. "A-1" represents player 1 in Team A, and "B-1" and "B-2" represent players 1 and 2 in Team B, respectively.

Team A – Turn 2 3:54 – 5:50	 Turn 90 – Move 3 forward – If on green, pick a Clover Card – Clover Card: Move 3 steps diagonally in any direction	A-1: I'm going to turn 90 degrees, move 3 steps forward, if I land on the green thing, and then, here me out guys. This is actually going to be crazy. B-1: (Moving Team A's character) Turn 90 degrees, move forward… A-1: One, two, three… B-2: No, it's the other way. A-1: It doesn't matter. Hey, I'm not done. B-1: (Counts on the board.) Move one, two, three, okay. A-1: (Counting on the board.) Okay. B-2: Are you done yet? Turn another 90 degrees. A-1: (Counts on the board again.) And then I'm going to play my Clover Card, move 3 steps diagonally in any direction. (Moves the ggame piece on the board following the code.) B-2: Is that it? A-1: Yeah, that was my turn.
Team B – Turn 2 5:51 – 7:24	 Turn 90 – Move 4 forward – Move 3 forward – If on yellow, pick a Clover Card – Move 1 forward – If on red, pick a Clover Card	B-1,2: (Looks through their tiles and discusses their strategy quietly.) B-2: We need to get another Clover Card. B-1: (Lays down a Clover Card.) B-2: Alright. Move forward to the right 2 steps, so we'd be here? B-1: (Removes the Clover Card.) Wait, actually, I'm not doing that. B-2: Well, we didn't hit "Go", so, [it's fine]. B-1, 2: (Counts on the board while discussing quietly. Puts down the code tiles in sequence.) B-2: And then do we have an orange card? B-1: Yes. Boom. Then, we'll be right here. B-2: We'd be facing this way (to the right). B-1: (Counts and moves the game piece on the board.) Wait, wait, could we do that? (Adds the last two code tiles. Taps the "Go" button.) "Go"!

Initial analyses for these code patterns involved calculating descriptive statistics, including means and standard deviations, as well as displaying those statistics using a line graph to visually display the participants' overall use of different code patterns. To determine the suitability of subsequent statistical tests, normality tests were conducted for each pair of days (Days 1 to Day 3) using the Shapiro-Wilk test. Results indicated that the differences for Code

Pattern 3 between Day 2 and Day 3 were not normally distributed (p = .012), while all other code patterns showed normal distribution across the comparisons. Consequently, the analysis employed the Wilcoxon signed-rank test, a non-parametric test suitable for paired data regardless of normality (Gibbons & Chakraborti, 2014). All statistical calculations in this study were conducted using SPSS software.

Pre- and Posttest

Pre-test and posttest assessments were conducted to measure the impact of the intervention on students' programming knowledge. The boys' and girls' pre- and posttest scores for all five items combined were normally distributed, which allowed for the use of paired sample t-tests for overall comparisons. However, the Shapiro-Wilk test revealed that several individual test items did not follow a normal distribution. Specifically, this was the case for items 1, 2, 4, and 5 when considering both genders combined; items 1, 3, 4, and 5 for males; and item 2 for females. Consequently, the Wilcoxon signed-rank test, a non-parametric method, was employed for these individual comparisons.

In this exploratory study, strict adjustments for multiple comparisons were not implemented. While some may suggest performing such corrections to avoid false-positive results, this decision aligns with the argument that these adjustments, though crucial in confirmatory studies aimed at changing established practice, are less critical in exploratory research (Althouse, 2016; Rothman, 1990). As highlighted by Althouse (2016), enforcing stringent correction standards in exploratory studies may oversimplify the nuanced issue of multiple comparisons, potentially stifling valuable initial findings. Instead, the exploratory nature of this study is clearly stated, with an acknowledgment of the need for subsequent research with pre-planned hypotheses to confirm the results.

Survey

Survey responses were analyzed using both quantitative and qualitative approaches. To investigate students' gameplay experience scores by gender, descriptive statistics were calculated to examine levels of enjoyment and interest over three days. The open-ended responses were analyzed using thematic analysis, following Braun and Clarke's (2006) six-phase approach. This process involves a systematic method of identifying, analyzing, and reporting patterns or themes within the data. It begins with familiarization with the data, where the researcher thoroughly reads and re-reads the data to become immersed in it. The next step is generating initial codes, which involves systematically coding notable aspects of the data throughout the entire dataset. Following this, the codes are collated into potential themes. These themes are then repeatedly reviewed and refined to ensure they accurately represent the data. The final phase involves weaving together the narrative of the analysis, supported by data extracts, to present a coherent and persuasive account of the findings.

Building on this approach, the data were systematically coded and then organized into broader themes that reflected the underlying patterns within the students' responses. For example, as visualized in Table 4.1, a response like "Having to work with what you have" was initially coded as "Optimizing Resources," while responses such as "Using strategy and my brain knowledge to achieve a goal" and "Having to problem-solve and be smart" were coded as "Strategizing" and "Problem-solving," respectively. During the thematic analysis process, these initial codes were reviewed and refined, ultimately being integrated into the broader theme of "Strategizing." This consolidation occurred because all these aspects—resource management, strategic planning, and problem-solving—were fundamental components of a broader strategic approach that students used during gameplay. By categorizing them under "Strategizing," the coding captured the essence of the students' tactical thinking and decision-making processes as they interacted with the game.

Similarly, responses such as "It helped me get better at coding," initially coded as "Learning Coding," and "I understand more," initially coded as "Gameplay Understanding," were reviewed and then consolidated into the broader theme of "Skill Improvement." This reflects the intertwined nature of coding and gameplay in the context of coding board games. Distinguishing between improving at coding and improving at the game itself became less meaningful, as the skills required for proficiency in the game were inherently related to coding proficiency. Therefore, both aspects were captured under the broader theme of "Skill Improvement," representing the students' overall growth and enhanced understanding through their interaction with the game. Following this process, Table 4.2 presents the final codes derived from the thematic analysis.

In addition to the qualitative analysis, the frequency of each code was also counted to determine which categories were most mentioned by each gender. This quantitative analysis of coding frequency provided further insights into the predominant themes in the responses of boys and girls, shedding light on the different aspects of gameplay and interaction that were emphasized by each gender.

Table 4.1: Examples of the coding process for student responses.

Student Response	Initial Coding	Final Coding
"Having to work with what you have."	Optimizing Resources	Strategizing
"Strategy"; "Strategies"	Strategizing	
"Using strategy and my brain knowledge to achieve a goal."	Strategizing	
"Having to problem-solve and be smart."	Problem-solving	
"Problem-solving"	Problem-solving	
"It helped me get better at coding and helped me understand it more."	Learning Coding	Skill Improvement
"I got better."	Skill Improvement	
"I understand more."	Gameplay Understanding	

Table 4.2: Coding scheme for qualitative analysis of student responses in gameplay.

Code	Definition
Strategizing	Responses where students explicitly mentioned "strategy" or "strategies" or described their approach to gameplay, including problem-solving and overcoming challenges.
Entertaining	Responses that emphasized the fun or entertaining aspects of the game; this often included words such as "fun" or "entertaining."
Teamwork	Responses focusing on the collaborative and social elements, including direct mentions of "teamwork" or enjoyment of playing with others.
Skill Improvement	Responses related to a deeper understanding of coding, learning new aspects, or feelings of increased competence.
Game Mechanics	Responses discussing the game's structure and rules, including its simplicity, pace, chance, characters, and overall mechanics.
Clover Cards	Responses specifically mentioning "Clover Cards," a distinct game feature that combines elements of chance with strategic play.
Rewards	Responses pertaining to the earning of in-game coins.
Limited Control	Responses expressing frustration over limited choices in selecting code tiles or feeling "unlucky" (e.g., other teams getting more favorable Clover Cards).
Opponent Team	Responses with complaints about interactions with the opponent team, such as arguing or rude comments.
Hindrance	Responses mentioning setbacks caused by the opponent team, like being forced to move on the board or giving away a coin.
Waiting	Responses noting having to wait for the other team or the game being “slow-paced.”
Winning, Losing, Coding, Competition	Direct mentions of these aspects were coded accordingly.
Everything	Assigned to a response explicitly stating "everything," indicating broad appreciation for all game aspects.

Interviews

The interview data were analyzed using the same thematic analysis approach applied to the survey data, as outlined by Braun and Clarke (2006). The audio-recorded interviews were transcribed, with key segments highlighted and annotated to capture initial insights and observations. Then, the data were systematically coded to identify significant features across the dataset. The primary focus was on aligning these codes with those already generated from the survey data, allowing for a comparison that added depth and context to the survey findings. Meanwhile, a new theme emerged from the interview data. Several interviewees expressed concerns about transitioning to programming on the computer despite their increased confidence in coding within the game. These concerns were captured with codes such as "Technology Reliability Concerns," "Paper-Based Preferred for Ease of Use," and "Expected Confusion," which were then categorized under the broader theme of "Technology-Related Anxiety." This new theme provided additional insights into the participants' experiences. All themes, both existing and new, were carefully reviewed and refined to ensure they provided a coherent and comprehensive understanding of the participants' experiences.

Ensuring Trustworthiness of Qualitative Data

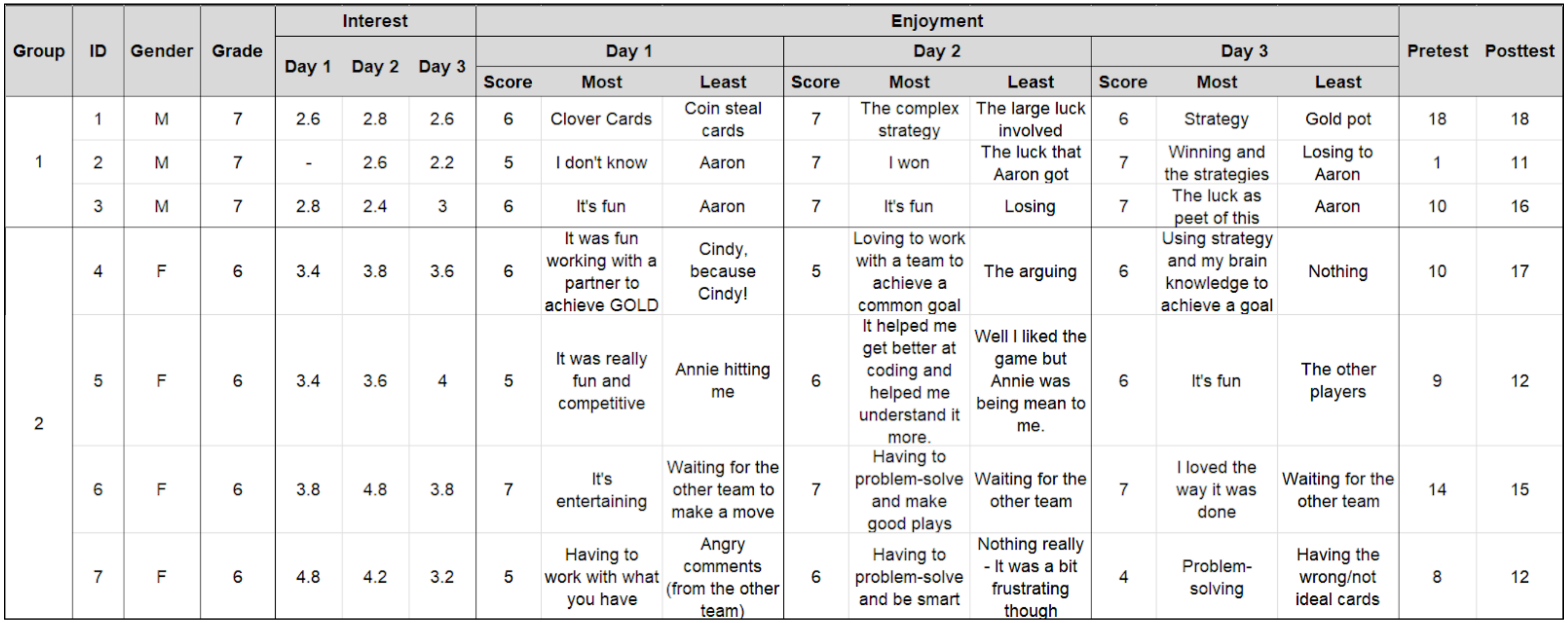
To ensure the trustworthiness of the qualitative data in this study, I employed data triangulation by integrating multiple sources of data, including gameplay observations, open-ended survey responses, interviews, and quantitative data such as pre- and post-test scores and gameplay metrics. This triangulation allowed for cross-verification of findings, providing a comprehensive understanding of students' engagement and learning processes (Lincoln & Guba, 1985; Patton, 1999). Additionally, I incorporated rich, thick descriptions by offering detailed accounts of the gameplay sessions, such as how students' use of coding strategies evolved and their reflections on these activities. These in-depth descriptions allowed readers to gain a clear understanding of the study's context and participants, enabling readers to evaluate the relevance of the research and its transferability to other contexts (Lincoln & Guba, 1985).

Integration of Quantitative and Qualitative Data

To integrate the quantitative and qualitative data, both types of data were collected concurrently, with qualitative data embedded within the quantitative framework. The quantitative component, which included paired sample *t*-tests and Wilcoxon Signed-Rank Tests, provided a comprehensive analysis of code pattern usage and changes in programming skills over time. Qualitative observations and interviews were conducted to gather contextual insights into students' gameplay strategies, engagement, and challenges. The qualitative data were analyzed to explain and illustrate the trends identified in the quantitative analysis. For example, qualitative observations provided detailed accounts of students' use of coding strategies during gameplay and interviews offered in-depth explanations of students' thought processes and experiences. Integrating quantitative and qualitative methods provided a well-rounded perspective on how the educational game influenced students' CT skills, merging analytical data with contextual observations.

To analyze the quantitative and qualitative data, a joint display was constructed to synthesize the findings. The joint display was an electronic spreadsheet that contained each team's pre- and posttest results, open-ended survey responses, daily gameplay statistics, and key themes from interviews and surveys. The creation and use of the joint display enabled a detailed and comprehensive analysis, allowing for an examination of both individual and collective trends observed in the study (Creswell & Plano Clark, 2017). It facilitated an understanding of how quantitative and qualitative data together highlighted consistent patterns or revealed unique aspects of the gameplay experience. Figure 4.5 shows a partially represented example of the joint display.

Examples of the joint display's utility were seen in its ability to track changes in interest and enjoyment ratings alongside qualitative comments over the course of three days. For instance, Student 3 consistently rated both interest and enjoyment high, with positive comments such as "It's fun" and "Winning and the strategies," indicating a strong alignment between quantitative and qualitative data. Furthermore, the joint display facilitated an overall comparison between genders. For example, both genders often mentioned the opposite team as the least enjoyed aspect, but for different reasons. Boys expressed frustration with losing or feeling less “lucky,” while girls were more concerned with the aggressiveness or “mean comments” from the opposing team.



Group	ID	Gender	Grade	Interest			Enjoyment									Pretest	Posttest
				Day 1	Day 2	Day 3	Day 1			Day 2			Day 3				
							Score	Most	Least	Score	Most	Least	Score	Most	Least		
1	1	M	7	2.6	2.8	2.6	6	Clover Cards	Coin steal cards	7	The complex strategy	The large luck involved	6	Strategy	Gold pot	18	18
	2	M	7	-	2.6	2.2	5	I don't know	Aaron	7	I won	The luck that Aaron got	7	Winning and the strategies	Losing to Aaron	1	11
	3	M	7	2.8	2.4	3	6	It's fun	Aaron	7	It's fun	Losing	7	The luck as peet of this	Aaron	10	16
2	4	F	6	3.4	3.8	3.6	6	It was fun working with a partner to achieve GOLD	Cindy, because Cindy!	5	Loving to work with a team to achieve a common goal	The arguing	6	Using strategy and my brain knowledge to achieve a goal	Nothing	10	17
	5	F	6	3.4	3.6	4	5	It was really fun and competitive	Annie hitting me	6	It helped me get better at coding and helped me understand it more.	Well I liked the game but Annie was being mean to me.	6	It's fun	The other players	9	12
	6	F	6	3.8	4.8	3.8	7	It's entertaining	Waiting for the other team to make a move	7	Having to problem-solve and make good plays	Waiting for the other team	7	I loved the way it was done	Waiting for the other team	14	15
	7	F	6	4.8	4.2	3.2	5	Having to work with what you have	Angry comments (from the other team)	6	Having to problem-solve and be smart	Nothing really - It was a bit frustrating though	4	Problem-solving	Having the wrong/not ideal cards	8	12

Figure 4.5: Example of the joint display of collected data. This figure displays the first 14 columns of the matrix. An additional 21 columns contain the pre- and posttest scores for each test item, daily counts of each code pattern used, and key themes from interviews and surveys. Names have been changed to pseudonyms to ensure participant confidentiality.

Results

In the following section, the results are organized and presented by research questions.

RQ1: What were the general trends in gameplay?

Trends in Code Pattern Usage

Table 4.3 and Figure 4.6 display the average instances and trends of the four distinct code patterns (CP) used during gameplay by each team over a three-day period. CP 1 (Movements Only) increased in usage from Day 1 to Day 2, followed by a decrease on Day 3. CP 2 (Conditional with Movements) consistently decreased over the three days. CP 3 (Loops with Movements) slightly increased from Day 1 to Day 3, and CP 4 (Loops and Conditional with Movements) showed an initial increase from Day 1 to Day 2, then a decrease on Day 3.

Table 4.3: Descriptive statistics of code pattern usage over three days (n = 32).

Code Pattern	Days	Mean	*SD*
1: Movements Only	1	1.63	1.77
	2	2.75	1.28
	3	1.00	1.31
2: Conditional with Movements	1	3.25	1.49
	2	2.50	1.51
	3	2.13	0.99
3: Loops with Movements	1	0.63	0.74
	2	0.38	0.52
	3	0.50	0.54
4: Loops and Conditional with Movements	1	0.50	0.93
	2	1.13	0.84
	3	0.87	0.84

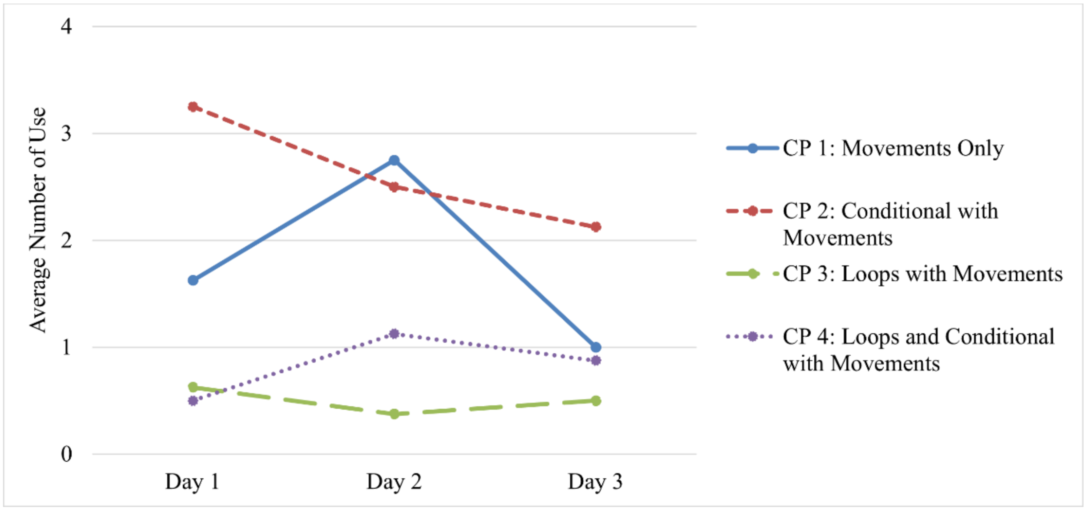


Figure 4.6: Trends in the average use of coding patterns over three days.

To analyze the data further, the Wilcoxon signed-rank test was used to compare the three days. Significant differences were found in two cases:

- CP 1 (Movements Only): A significant decrease was observed from Day 2 to Day 3 (*Z* = -2.20, *p* = .028).
- CP 2 (Conditional with Movements): A significant decrease was observed from Day 1 to Day 3 (*Z* = -2.26, *p* = .024).

No other significant differences were observed in the comparisons. Detailed results are presented in Table 4.4.

Table 4.4: Wilcoxon signed-rank test results for code pattern usage.

Code Pattern	Pair	Mean Difference	*SD*	*Z*	*p*
1: Movements Only	Day 2—Day 1	1.13	2.42	-1.42	.156
	Day 3—Day 1	-0.63	2.39	-0.74	.461
	Day 3—Day 2	-1.75	1.58	-2.20	.028*

2: Conditional with Movements	Day 2—Day 1	-0.75	1.67	-1.19	.236
	Day 3—Day 1	-1.13	0.99	-2.26	.024*
	Day 3—Day 2	-0.38	1.41	-0.75	.453
3: Loops with Movements	Day 2—Day 1	-0.25	1.04	-0.71	.480
	Day 3—Day 1	-0.13	0.99	-0.38	.705
	Day 3—Day 2	0.13	0.64	-0.58	.564
4: Loops and Conditional with Movements	Day 2—Day 1	0.63	1.30	-1.39	.163
	Day 3—Day 1	0.38	1.41	-0.75	.453
	Day 3—Day 2	-0.25	0.71	-1.00	.317

* $p < .05$

Gameplay Observation Analysis

Following the quantitative findings, a qualitative analysis of the observation data was performed to understand the factors contributing to the different trends in code pattern usage over the three days and to examine nuanced trends that were not evident in the quantitative data alone.

The general trend identified in the quantitative analysis was reflected in students' movements on the gameboard across three days. Figure 4.7 illustrates the gameboard paths taken by Groups 1 and 3 as representative examples for each day. On Day 1, both groups primarily explored the gameboard through multiple horizontal movements, focusing on collecting nearby coins. Most of their movements occurred within the rainbow area, which explains the high frequency of CP 2 (Conditionals with Movements) used on Day 1 (see Figure 4.6), as they frequently used color-conditionals to try out what they could do with Clover Cards.

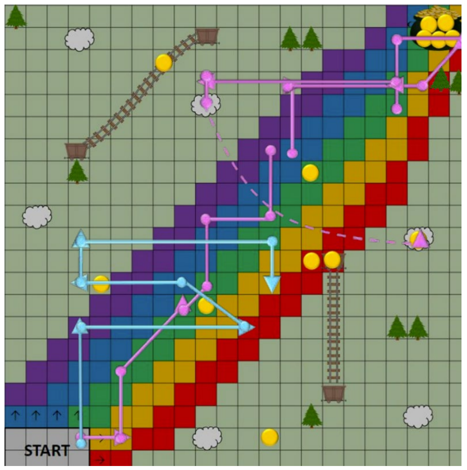
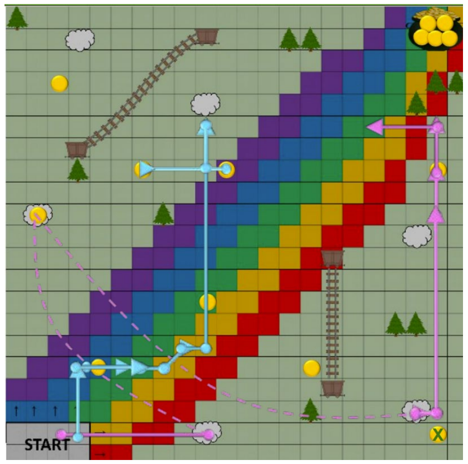
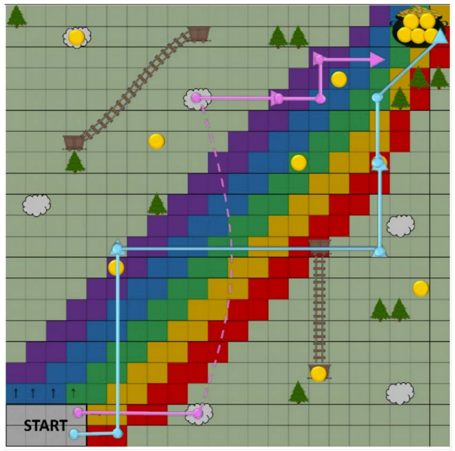
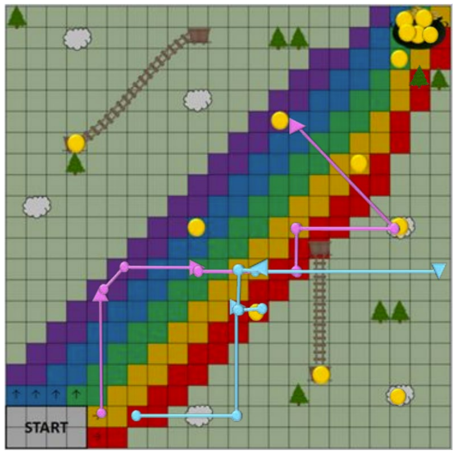
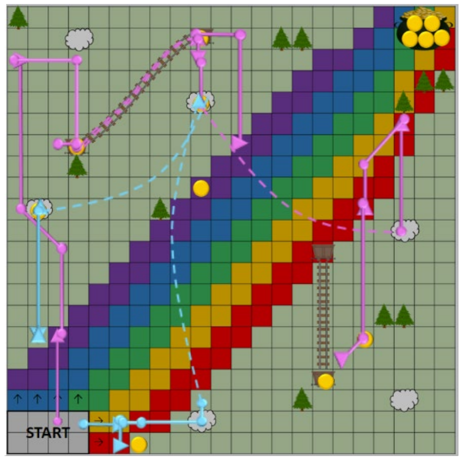
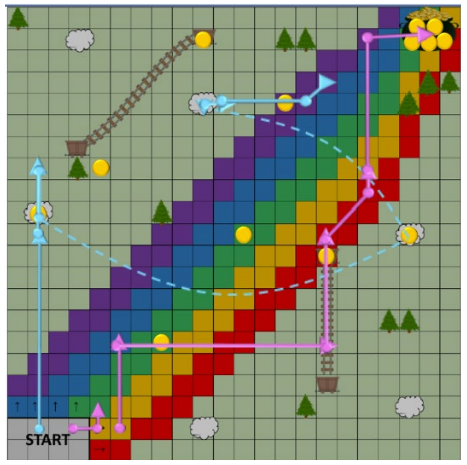
Group	Day 1	Day 2	Day 3
1			
3			

Figure 4.7: Comparative gameboard paths by Groups 3 and 1 across three days. Pink lines represent Team A, and light blue lines represent Team B. Numbers in circles indicate the turns taken by Team A, while numbers in squares indicate the turns taken by Team B.

On Day 2, Group 3 showed more active and varied movements across the board, including multiple cloud jumps, as indicated by the dotted lines in Figure 4.7. Meanwhile, Group 1's movements were more refined and purposeful, directing their path toward the gold pot while collecting coins along the way. However, their movements per turn remained relatively short compared to Day 3. Unlike on the first day, both groups moved beyond the rainbow area and used more cloud-conditionals. Notably, Figure 4.7 shows that a team in both groups (Team A in Group 1 and Team B in Group 3) used the cloud-conditional twice in a single turn, indicating their use of conditional combined with loops. This pattern aligns with the increased average number of CP 1 (Movements Only) and CP 4 (Loops and Conditionals with Movements), while CP 2 (Conditionals with Movements) decreased.

On Day 3, both groups significantly reduced the total number of moves, with each turn involving longer and more efficient actions; both groups managed to collect five coins in fewer turns than previous days. This pattern reflects the overall decrease in the use of code patterns, with the usage of those involving loops (CP 3 and CP 4) remaining relatively unchanged.

A closer look at the codes students created during the gameplay sessions revealed a progression consistent with the quantitative data. Overall, students began using CP 1 (Movements Only) and CP 2 (Conditionals with Movements) less frequently while becoming more sophisticated with their use of loops (CP 3) over the three days. The patterns observed in Groups 4 and 1 illustrate this development (see Table 4.5).

Table 4.5: A comparative illustration of using loops with movements in Group 4's gameplay over three days. Loops are highlighted in bold for easier identification.

	Day 1	Day 2	Day 3
Group 4			
Codes	**Repeat below 2 times** Move 1 forward	Turn 180 **Repeat below 2 times** Move 3 forward	**Repeat below 2 times** **Repeat below 3 times** Move 1 forward
Description	A loop tile was used to repeat the actions of moving forward.	A loop tile was used to repeat the actions of moving forward following the turn and move tiles in a specific sequence.	A nested loop was used to repeat the actions of moving forward.
Group 1			
Codes	**Repeat below 3 times** Move 2 forward Move 1 forward	Move 1 forward Turn 90 Move 2 forward **Repeat 2 times below** Move 4 forward	**Repeat 2 times below** Move 2 forward Turn 90
Description	A loop tile was used to repeat the actions of moving forward.	A loop tile was used to repeat the actions of moving forward following the turn and move tiles in a specific sequence.	A loop tile was used to repeat the combined actions of moving forward and turning.

On Day 1, Group 4 used a loop to double their steps with no additional use of loops. On Day 2, they used a loop along with a turn tile, creating a longer sequence. On the final day, they employed a nested loop, generally considered a more advanced concept than a single loop (Yamashita et al., 2016), to multiply their movement from one to six steps.

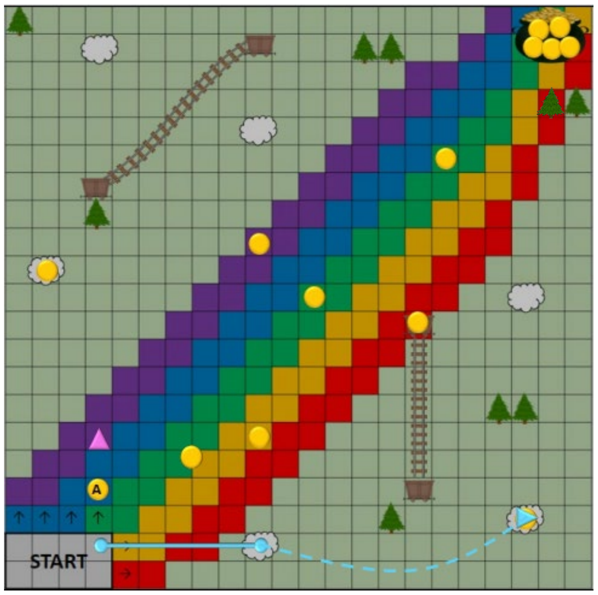
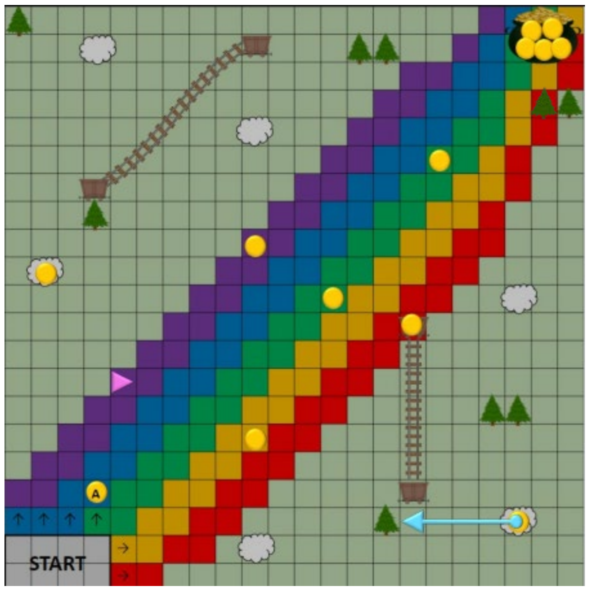
A similar advancement was observed in Group 1's codes from Days 1 to 2. On the second day, Group 1 extended the length of the action of moving forward and used it along with other code tiles (as did Group 4). On Day 3, Group 1 used the loop tile to repeat the combined actions of moving forward and turning. These patterns suggest that students’ skills developed progressively, where each day, more advanced looping skills were employed that were not seen in previous sessions.

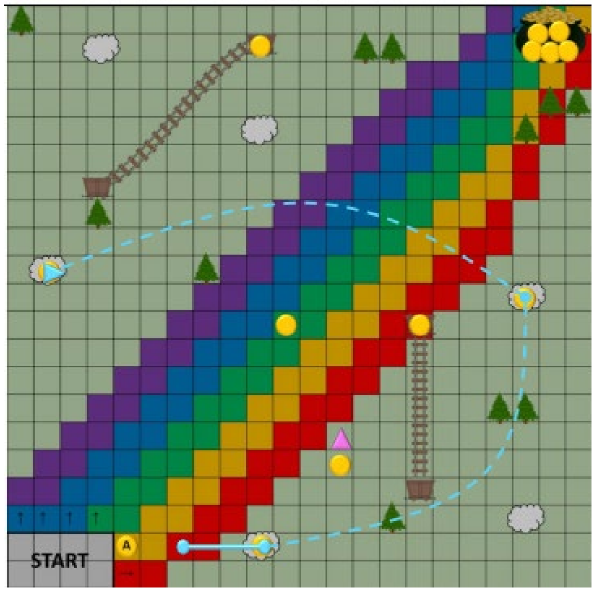
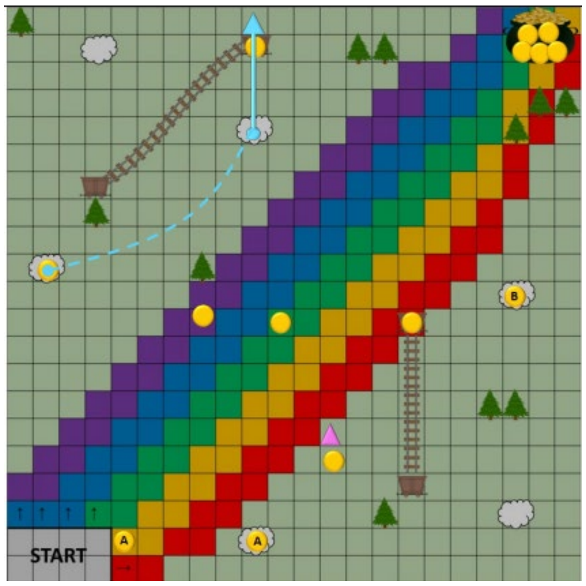
In addition to using more complex loops, students' use of loops in combination with conditionals evolved over the three days, leading to greater efficiency in collecting rewards with fewer moves. Figure 4.8 shows a representative example of the use of loops and conditionals by Team B in Group 2 (tracked by light blue lines) across three days. Specifically, it displays two consecutive game turns together to provide a clearer context for the team’s strategic movements, as they are often closely linked to form a unified approach.

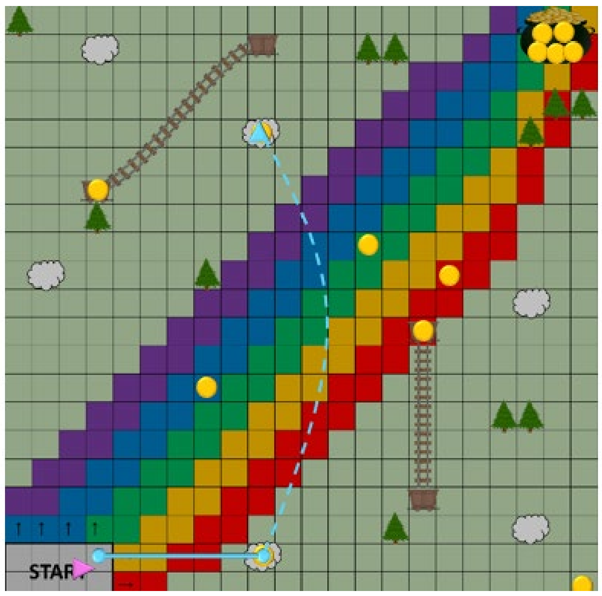
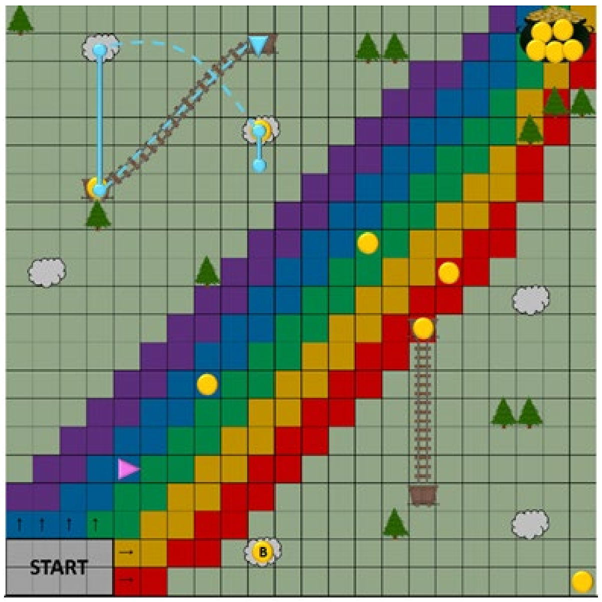
Team B of Group 2 demonstrated how the participants’ use of loops and conditionals became more sophisticated over three days. On the first day, Team B used a cloud-conditional tile to move to a new cloud and collect a coin. In the following turn, they navigated back towards the mine cart, aiming to secure another coin at its opposite end. This strategy, however, required an additional four turns to acquire the second coin. On the second day, the team sought to obtain multiple coins more efficiently. To do that, they employed a cloud-conditional tile to collect multiple coins while moving closer to the pot. This strategy ultimately enabled them to reach the pot of gold first, winning the round.

On the third day, Team B used a cloud conditional from their first move to collect multiple coins and approach the gold pot. The recording of their conversation revealed that they had a specific plan to move towards the upper minecart (as they did on Day 2). However, the opposing team’s use of a Clover Card forced them to move backward one step, to which the team vocally expressed their frustration with their original plan being disrupted. In the following turn, however, the team realized another opportunity. After a lengthy discussion, they used another cloud conditional to put themselves in a position to move towards a mine cart, resuming their original plan while securing an additional coin.

Figure 4.8: An illustration of the use of loops and conditionals in Group 2's gameplay over three days. The colors in the figure represent the respective teams—pink for Team A and light blue for Team B. The alphabet on the coins indicates the team that had gotten the coin.

Day 1	
	
Repeat below 2 times Move 3 forward If on a cloud, transport to another cloud	Turn 180 Repeat below 2 times Move 2 forward
A loop tile was used to repeat the action of moving forward. A cloud-conditional was used to collect a coin.	On the following turn, a loop tile was used to repeat the actions of moving forward toward the mine cart with the aim of gaining another coin at the other end of the cart.

Day 2	
 Move 3 forward Repeat below 2 times If on a cloud, transport to another cloud A loop tile was used to repeat the cloud-conditional tile to collect 2 coins at once.	 If on a cloud, transport to another cloud Turn 90 Move 4 forward Another cloud-conditional tile was used to collect another coin.
Day 3	

	
Move 3 forward Move 3 forward Repeat below 2 times If on a cloud, transport to another cloud Turn 90	Move 1 forward If on a cloud, transport to another cloud Turn 90 Turn 90 Move 5 forward
A loop tile was used to repeat the cloud-conditional tile to collect two coins at once. Then, a turn was used to face toward the mine cart.	On the following turn of the opponent team, this team was moved backward one step. Instead of reaching toward the upper mine cart, which was their original plan, they decided to use a cloud conditional again, along with movement and rotation tiles, to collect an additional coin before returning to their desired spot.

RQ2: What were the changes in knowledge over time, and how did they differ by gender?

Pre- and Posttest Results

To assess pre- and posttest results, paired sample *t*-tests were conducted to compare students' overall test scores, and a Wilcoxon Signed-Rank Test was used to evaluate the scores for each individual test item. Sixteen out of 17 students' scores (ten boys and six girls) were included, as one female student was absent for the posttest. When comparing the overall scores, a paired sample *t*-test revealed statistically significant improvements from pretest to posttest across all groups. For all genders, the mean pretest score was 7.06 (*SD* = 4.86), and the mean posttest score was 11.44 (*SD* = 5.02), *t*(15) = -5.36, *p* < .001. For boys, the pretest mean was 7.10 (*SD* = 5.65), and the posttest mean was 11.60 (*SD* = 5.76), *t*(9) = -3.55, *p* = .006. For girls, the pretest mean was 7.00 (*SD* = 3.69) and the posttest mean was 11.17 (*SD* = 3.97), *t*(5) = -5.93, *p* = .002. These results are summarized in Table 4.6 and visually depicted in Figure 4.9, which compares the mean pretest and posttest scores by gender.

Table 4.6**:** Paired sample *t*-test results for pretest and posttest scores by gender.

		n	*M*	*SD*	*t*	*p*
All genders	Pretest	16	7.06	4.86	-5.36	< .001***
	Posttest	16	11.44	5.02		
Boys	Pretest	10	7.10	5.65	-3.55	.006**
	Posttest	10	11.60	5.76		
Girls	Pretest	6	7.00	3.69	-5.93	.002**
	Posttest	6	11.17	3.97		

** *p* < .01. *** *p* < .001

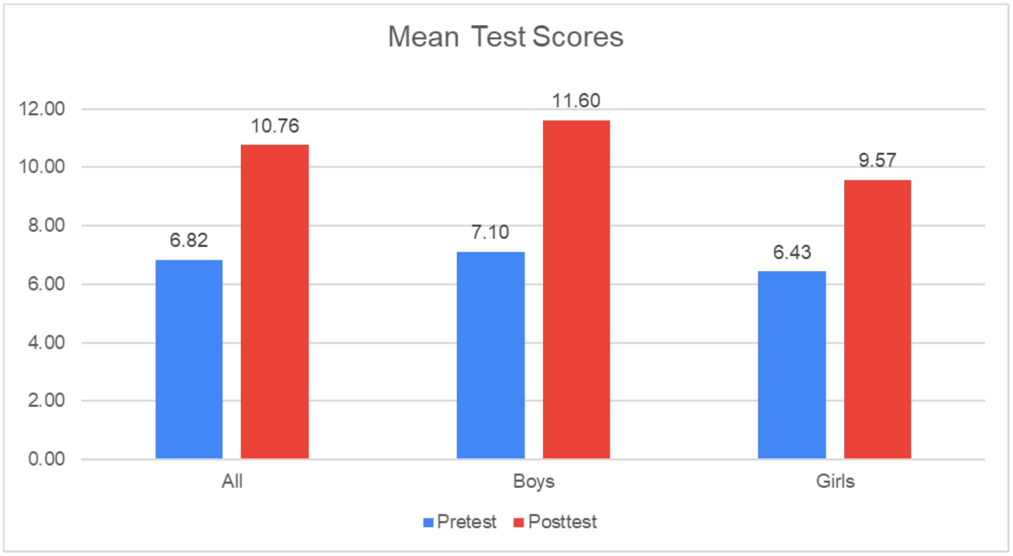


Figure 4.9: Comparison of mean pretest and posttest scores by gender.

Next, the Wilcoxon Signed-Rank Test was conducted to evaluate whether there was a statistically significant difference in student performance on the five different test items (i.e., Sequencing, Loops and Conditionals, Loop Efficiency, Advanced Efficiency, and Debugging) before and after the gameplay sessions. Table 4.7 displays the results comparing overall differences in pre- and posttest mean scores and differences by gender. The Wilcoxon Signed-Rank Test indicated statistically significant improvements in posttest scores in Loop Efficiency ($Z = -2.36$, $p = .019$), Advanced Efficiency ($Z = -2.62$, $p = .009$), and Debugging ($Z = -2.43$, $p = .015$). There were no significant changes in scores for Sequencing ($Z = -1.81$, $p = .070$) and Conditionals and Loops ($Z = -0.88$, $p = .380$).

Table 4.7: Wilcoxon signed-rank test results for pretest and posttest scores by item.

	Overall (n = 16)		Boys (n = 10)		Girls (n = 6)	
	M(*SD*)	*p*	*M*(*SD*)	*p*	*M*(*SD*)	*p*

	Item	Pre	Post		Pre	Post		Pre	Post	
1	Sequencing	1.94 (1.39)	2.50 (1.46)	.070	1.90 (1.66)	2.30 (1.49)	.257	2.00 (0.89)	2.83 (1.47)	.157
2	Conditionals and Loops	2.38 (1.78)	2.56 (1.59)	.380	2.60 (1.58)	2.80 (1.32)	.516	2.00 (2.19)	2.17 (2.04)	.317
3	Loop Efficiency	1.44 (1.63)	2.56 (1.63)	.019*	1.10 (1.79)	2.30 (1.83)	.071	2.00 (1.26)	3.00 (1.26)	.131
4	Advanced Efficiency	1.00 (1.59)	2.38 (1.41)	.009**	1.00 (1.63)	2.60 (1.51)	.041*	1.00 (1.67)	2.00 (1.26)	.131
5	Debugging	0.31 (1.01)	1.44 (1.55)	.015*	0.50 (1.27)	1.60 (1.71)	.059	0.00 (0.00)	1.17 (1.33)	.102

* $p < .05$. ** $p < .01$

When examining the results by gender, the difference from the pre- to posttest for boys was statistically significant in Advanced Efficiency ($Z = -2.04$, $p = .041$). No significant changes were found for Sequencing ($Z = -1.13$, $p = .257$), Conditionals and Loops ($Z = -0.65$, $p = .516$), Loop Efficiency ($Z = -1.81$, $p = .071$), and Debugging ($Z = -1.89$, $p = .059$). For girls, no significant changes in scores were revealed in any items: Sequencing ($Z = -1.41$, $p = .157$), Conditionals and Loops ($Z = -1.00$, $p = .317$), Loop Efficiency ($Z = -1.51$, $p = .131$), Advanced Efficiency ($Z = -1.51$, $p = .131$), and Debugging ($Z = -1.63$, $p = .102$).

RQ3: What were the patterns in attitude, and how did they differ by gender?

Enjoyment and Interest Scores

The survey asked students to indicate their level of enjoyment of the game and their interest level in learning coding, along with short written responses to explain their rankings. Figures 4.10 and 4.11 show the differing trends in enjoyment and interest between boys and girls. For boys (Figure 4.10), average enjoyment scores increased steadily over three days (Day 1: 6.10, Day 2: 6.35, Day 3: 6.50), while their interest in learning coding declined (Day 1: 3.75, Day 2: 3.00, Day 3: 2.70). For girls (Figure 4.11), both enjoyment and interest showed similar patterns: enjoyment scores rose from Day 1 (5.57) to Day 2 (5.83) but fell on Day 3 (5.00), mirroring their interest levels, which peaked on Day 2 but declined on the other days (Day 1: 3.63, Day 2: 4.33, Day 3: 3.67).

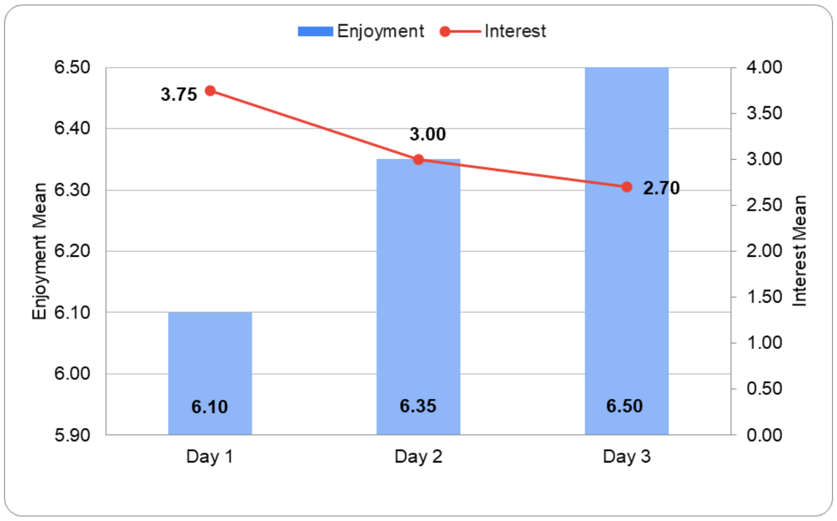


Figure 4.10: Boys' comparison of daily enjoyment and interest mean scores over a three-day period.

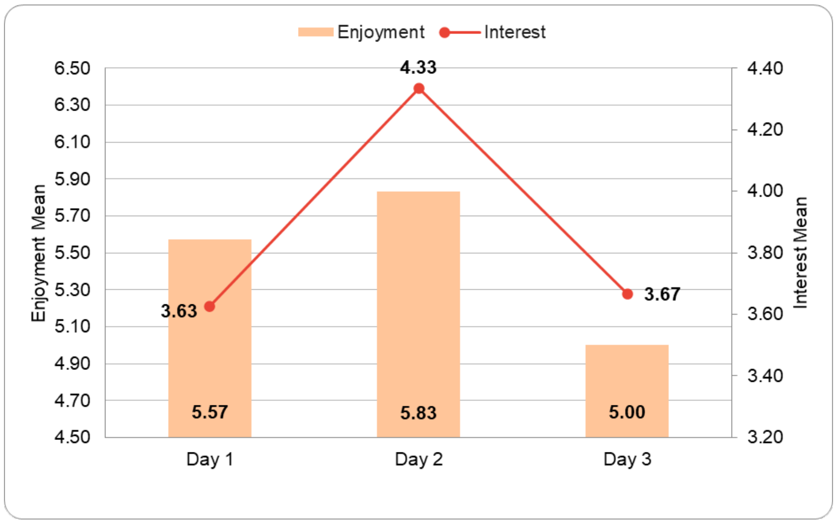


Figure 4.11: Girls' comparison of daily enjoyment and interest mean scores over a three-day period.

Open-Ended Survey Responses

The open-ended questions asked students to identify gameplay aspects they found most and least engaging. Each response was assigned a code based on the theme it represented, and the frequency of each code was counted separately by gender. The coded responses are detailed in Table 4.8. Boys showed a strong preference for “Strategizing,” which appeared in 8 responses, and also expressed notable interest in “Clover Cards” (5 responses), highlighting an inclination toward tangible strategic elements. Girls similarly preferred “Strategizing” (6 responses) and placed similar emphasis on “Teamwork” (5 responses). Boys predominantly disliked aspects of “Limited Control” (8 responses), followed by “Waiting” (4 responses) and “Hindrances” (4 responses), indicating a preference for greater autonomy and a faster-paced gameplay experience. Girls, on the other hand, expressed greater frustration over conflicts with the “Opponent Team” (8 responses), followed by “Limited Control” (5 responses) and “Losing” (3 responses), suggesting that they were relatively more sensitive to the social dynamics of the gameplay.

Table 4.8: Most and least enjoyed gameplay aspects by gender.

Most Enjoyed		Least Enjoyed	
Boys	Girls	Boys	Girls
8 Strategizing	6 Strategizing	8 Limited control	8 Opponent team
5 Clover cards	5 Teamwork	4 Waiting	5 Limited control
4 Entertaining	3 Entertaining	4 Hindrance	3 Losing
4 Game Mechanics	3 Winning	3 Coding	
2 Winning	2 Rewards	2 Opponent team	
1 Rewards	2 Coding	2 Losing	
1 Everything	1 Skill Improvement	1 Unchallenging	
1 Teamwork			

	1 Competition		

Interviews

The interviews offered a more nuanced understanding of the participants' survey responses. Both boys and girls expressed excitement about the strategic elements of the game. They found the use of loops and conditionals to be particularly exciting due to the benefit of being able to achieve more in fewer turns. For instance, when students were asked about their most memorable moments in the game, a pair of boys shared how they were able to "[get] so many Clover Cards at once," proudly elaborating that they were able to earn 12 Clover Cards at once using loops. Two girls in different groups similarly shared how their teams strategically used the repeat tiles in conjunction with movement and conditional tiles, which resulted in executing a large number of moves in a single turn. They enthusiastically described their approach as part of their "specific strategy" and an opportunity to "go crazy on" using the code tiles.

When comparing genders, all the male interviewees expressed their enjoyment of the strategic aspect of the game and what they were able to achieve. When interviewing girls, while they agreed on enjoying the strategizing aspect, many also included mentions of their peers in both positive and negative ways. For example, when a group of boys were asked what they enjoyed the most about the game, they were most excited about betting on luck to get "good cards" and then creating the best strategy, although sometimes they were frustrated when the luck was not in their favor. Other boys favored “Clover Cards and the clouds (cloud-conditionals)” and the balancing of strategy and luck. This enthusiasm for strategizing aligned with the quantitative finding that strategizing was favored among boys across all days of gameplay. When girls were asked what they liked most about the game, they also emphasized the aspects of strategizing and problem-solving. One student shared her enjoyment with using the cloud conditionals, as it made her feel like having “a weapon from Stranger Things.” Another student shared:

> I thought it was fun because you really have to think outside the box and problem-solve because there may be a spot that you turn to reach where you think that you don’t have the right cards to move there – but you actually do. You just have to figure them out in a different way.

However, the interviews also showed that girls were more sensitive to the social element of the game. A pair of girls shared how they enjoyed working as partners and emphasized the benefit of working in teams as the other person can offer a new perspective. Another girl mentioned, “It was really fun because I love teamwork and working with my friend Kim (pseudonym). It was really fun. Probably the thing that really kind of set me back was arguing about what to put down.” In contrast, no boys mentioned the social aspect unless they were specifically asked about their experience playing with their partners. When prompted, they mostly showed neutral preference, describing both the advantages and disadvantages, such as gaining new ideas but also being slowed down by having to discuss and explain strategies.

When students were asked about their interest in learning coding, many, regardless of gender, described coding on a computer as “interesting” but looks challenging or confusing. However, a subtle difference was observed in each gender’s perception of the practice of coding.

Male students often highlighted the intellectual aspect: some mentioned enjoying “getting better” at the game in terms of strategizing and utilizing the codes. Many also liked “feeling smart” when making satisfactory moves in the game, as reflected in one student’s comment, “It made me feel smart. I’m not very smart.”

When asked to imagine what coding on the computer would look like, boys described images such as “wearing cool glasses” while typing on a computer but showed reluctance to engage with it, seeing it as less enjoyable and different from playing the related game. For instance, one male student said, “To be honest, it seems really cool. But my friend says it sucks. He said it was too hard. He said that he thought it was too boring. But it sounds interesting.” Another student noted that while the game experience increased their confidence in learning coding, it still seemed very different from playing the game, mostly involving typing codes on the computer.

On the other hand, when girls described the practice of coding on the computer, all the female interviewees highlighted the challenges of computer-based learning. For example, one student mentioned, “I usually like (coding on) paper a lot more because (on) computers, it could die, or it could accidentally get deleted or something like that, and that doesn't really happen with paper or anything.” Other girls agreed that coding on the computer was “riskier” due to factors such as “glitches” or the computer turning off unintentionally. One described computer coding as “a permanent thing you could mess up really bad.” Many expressed their preference for using paper-based coding as it was safer and “easier to keep up with.”

Female students also reported frustration with receiving timely support when learning coding in a classroom setting:

I might not take a class in [coding]. I might just do it at home because I feel like coding

> classes seem really confusing … like having your computer always charged and always having so many people, so it's like [the instructors] can't just help you. If something doesn't work, you might not be able to do anything.

Another student agreed:

> It's like you want to [learn to code], but you don't want to do that at the same time because there's going to be a lot of people (in the class). But if you do it, … I feel like it could be if you do it at your house.

These data showcases the differing perceptions and preferences between male and female students regarding coding, revealing distinct challenges and experiences that each group encounters.

Discussion

The following section highlights how the study results address the research questions and discusses the broader impact of the findings on the understanding and teaching of CT through unplugged educational games. Through this exploration, the study provides insights and recommendations that may be valuable for future research and practice.

Advancement in Gameplay

Over the three days of gameplay, students increasingly utilized advanced concepts like loops and conditionals to achieve greater efficiency in their gameplay. From Day 1 to Day 2, students' use of "Movements Only" (CP1) and "Loops and Conditional with Movements" (CP4) increased, while their use of "Conditional with Movements" (CP2) and "Loops with Movements" (CP3) decreased. This trend likely reflects how students initially explored the different functions of movement, loops, and conditional code tiles separately, then, by the second day, became familiar with the game's basic mechanics and gained the confidence to explore the gameboard more actively, experimenting with different code patterns by combining loops and conditionals together with movements.

When comparing Day 1 to Day 3, there was an overall decrease in the use of all code patterns, with only a slight increase observed in CP 3. This decrease might be explained by the observation and posttest data suggesting that students became more efficient in their gameplay over time. By Day 3, students appeared to use more advanced and purposeful moves, such as using nested loops and strategic combinations of conditionals, enabling them to reach their goals in fewer turns and reducing the need for additional code patterns. This advancement aligns with the changes in pre- and posttests, where students generally showed a statistically significant increase in scores related to loop efficiency, advanced efficiency, and debugging.

This progression highlights the potential of educational games like *Lucky Codes* in providing students with an effective tool to develop CT skills. Through the gameplay of *Lucky Codes*, students had the opportunity to engage with CT concepts, such as sequencing, loops, conditionals, and debugging, in a manner that was both interactive and enjoyable. The positive impact of using board games to teach these concepts reflects many previous research findings, which have shown that such approaches can significantly enhance students' understanding of loops and conditionals, thus improving their computational thinking skills (Alamer et al., 2015; Ballard & Haroldson, 2021; Liu et al., 2022; Sun et al., 2021; Zhang et al., 2024).

Moreover, the game encouraged iterative learning and experimentation, allowing students to make mistakes and learn from them without the pressure of falling behind their peers or the fear of “messing up,” as some students mentioned in their interviews. Such an environment fostered active problem-solving processes, enabling students to try different strategies and discover more efficient solutions on their own. This aligns with other research studies that found similar benefits in game-based learning environments, where students exhibited improved problem-solving skills and a deeper understanding of CT concepts through iterative learning and trial-and-error approaches (Brennan & Resnick, 2012; Grover & Pea, 2013).

Additionally, the findings support the game's design of incentivizing the use of conditionals and loops to be effective in encouraging students to actively utilize the targeted skills beyond simple pathfinding movements. For example, utilizing loops and conditionals was highly attractive in the game as they enabled large moves at once (by repeating moves or cloud jumps) and the acquisition of multiple Clover Cards at once (by repeating the color-conditional tiles) that could not be achieved otherwise and provided an easier way to earn rewards. The random draw aspect of Clover Cards added to the excitement of utilizing more color-conditionals combined with loops. These elements of the game encouraged players' deeper engagement with the targeted CT skills. This design approach addresses the limitations of some existing coding board games, which often feature limited programming elements, particularly regarding loops (Scirea & Valente, 2020; Wu, 2018). By integrating these incentives, the game appeared to make conditionals and loops more engaging and accessible, potentially reinforcing their importance in developing efficient problem-solving strategies.

Gender Differences in Gameplay Preferences

Further analysis of gender differences in knowledge gain and attitudes offered a deeper understanding of the general gameplay trends, revealing distinct patterns in how male and female students interacted with and benefited from the game. Boys showed a clear preference for the competitive and strategizing aspects, particularly enjoying the successful execution of strategies and engaging with multiple Clover Cards to win the game. Their dislike of "Limited Control," "Waiting," and "Hindrances" emphasized their desire for gameplay that maximizes direct action and autonomy, which are key components of a competitive gaming experience (Przybylski et al., 2010; Ryan et al., 2006). This aligns with broader educational research suggesting that male students often favor environments where they can test and improve their strategic thinking against peers (Kinzie & Joseph, 2008; López-Fernández et al., 2021). Boys demonstrated larger improvement on the posttest than girls, especially in their ability to use conditionals and loops for advanced efficiency. This result highlights the potential for competitive and strategic game elements to enhance boys' engagement and learning outcomes in CT. A similar result was also found in Admiraal et al.'s (2014) study of digital games, where boys' competing with other teams was positively related to their performance than girls.

In contrast, while girls also reported enjoying the strategic aspects of the game, they placed a greater emphasis on the social dynamics of gameplay than boys, valuing collaboration, teamwork, and communication. Interview and survey responses of this study supported these findings, as they frequently mentioned their peers and teamwork in both positive and challenging contexts that involved collaborative decision-making and problem-solving. This preference for social interaction aligns with existing studies that emphasize the significant role of social dynamics in girls' engagement in educational activities, suggesting that girls often find motivation and satisfaction through collaborative efforts and shared experiences (Crowe, 2003; Hartmann & Klimmt, 2006; Niederle & Vesterlund, 2010; Zachopoulou et al., 2004).

Gender-Specific Trends in Interest and Enjoyment

Gender-based differences in coding interest and enjoyment trends emerged during the gameplay sessions. In terms of interest in learning coding, boys initially exhibited a slightly higher interest compared to girls, which reflects findings from earlier studies that male students generally had a higher interest in or positive attitudes toward CS-related subjects (Kong et al., 2018; Sun et al., 2021; Witherspoon et al., 2016). As the game sessions progressed, boys' interest gradually decreased, whereas girls’ interest increased on Day 2 and decreased on Day 3. The overall decreased interest in coding observed in both genders was similar to the findings of Hong et al.'s (2016) study with a digital game, where there was a gradual decrease in students' interest in the learning activity despite improved performance. The reason for this general trend could be attributed to the game's task becoming less challenging as the students become better at it, which could be supported by Chen et al.’s (2023) finding that if the task can be completed quickly, the effectiveness of unplugged learning of CT may diminish, therefore suggesting the learning activity should increase task difficulty with extended duration.

However, the patterns in enjoyment levels between boys and girls revealed notable gender differences. Boys’ enjoyment gradually increased despite their declining interest in coding. This could suggest that boys’ enjoyment level was related to their proficiency in the game, but they were more likely to separate the act of playing the educational game from learning. Their general preference for competitive and strategic elements in digital games appeared to carry over into the unplugged gaming context when similar features were present. In contrast, girls’ enjoyment level mirrored their interest in coding: both increased on Day 2 but dropped on Day 3. This could suggest that girls may have linked the educational game more closely with the learning process. However, the game’s appeal to strategizing and competition was insufficient to sustain their enjoyment. As boys became more competitive, girls’ lower preference for competition seemed to negatively impact their overall enjoyment of the game. This observation aligns with Hartmann and Klimmt's (2006) study, which found that female participants were less attracted to the competitive elements of digital games, indicating that similar dynamics are present in both digital and unplugged gaming contexts.

Implications

The findings from this study offer valuable insights into how educational games can be designed and implemented to enhance students' CT skills. By addressing gender-specific preferences and maintaining student engagement, these implications provide practical suggestions for educators and game designers to optimize the learning experience.

Integrating unplugged educational games into the curriculum can provide an engaging, low-risk environment for students to explore CT concepts. By offering opportunities for both individual strategic thinking and teamwork, these games can accommodate the diverse learning preferences of male and female students. Furthermore, games that encourage iterative learning allow students to make and learn from mistakes without pressure, fostering active problem-solving skills. Incorporating adjustable difficulty levels can also help to maintain engagement and provide appropriate challenges as students advance in their skills.

While unplugged educational games offer many benefits, it's important to recognize that students' interest in the game may fluctuate over time. To maintain engagement, strategies such as gradually increasing task difficulty can help sustain both challenge and interest. Additionally, incorporating reflection sessions with the instructor can reinforce the connection between gameplay and the learning process, helping students to see the practical applications of the concepts they are exploring. This approach may be particularly beneficial for boys, who might otherwise separate the act of playing from the learning experience. By understanding the relevance and real-world applications of the skills acquired through gameplay, students are likely to find the learning process more engaging and meaningful, ultimately fostering a deeper interest in coding.

Limitations and Future Research

This study has several limitations that should be considered when interpreting the findings. One significant limitation is the small sample size, consisting of 17 students, which may not provide a fully representative view of the broader student population. Additionally, the uneven number of male and female participants (10 boys and 7 girls) may have influenced the statistical results related to gender differences. Another limitation lies in the pre- and post-tests, which were developed by the researcher and did not go through a rigorous validation process. While these tests effectively assessed students' skills in the context of the game, they did not specifically measure the transferability of those skills to broader programming scenarios. Finally, the lack of statistically significant results in some areas of the study suggests that it may not have had sufficient power to detect subtle differences or smaller effect sizes, which could be addressed in future research with larger and more balanced samples.

To address these limitations and build upon the findings of this study, future research should focus on several key areas. First, conducting studies with larger and more diverse samples from different schools would offer a more comprehensive understanding of the impact of educational board games on CT skills across different student populations. Additionally, exploring how students apply the coding skills acquired through the board game in a computer-based programming environment would offer valuable insights into the transferability of learning from unplugged to digital coding contexts. Subsequent studies could also incorporate pre-planned hypotheses to rigorously test and confirm the findings observed in this exploratory study. Addressing these areas could help overcome the limitations of the current study and contribute to the development of more effective and inclusive game-based learning tools for CT education.

Conclusion

This study sought to explore how boys and girls engage with CT skills through the educational board game *Lucky Codes*. The primary research questions focused on identifying general gameplay trends, assessing changes in CT knowledge over time, and examining patterns in students' attitudes, with attention to potential gender differences. The findings provide valuable insights into the dynamic learning processes that occur during gameplay and highlight the effectiveness of unplugged games in teaching CT concepts to young learners.

The study revealed distinct trends in gameplay, with students progressively utilizing more complex coding skills, specifically loops and conditionals, over the three days of gameplay. This progression highlights how the game facilitated a deeper engagement with CT concepts, allowing students to experiment, iterate, and refine their strategies in a low-risk environment. The pre- and posttest results further validated these observations, showing overall improvements in students' CT skills.

Gender differences emerged as an important aspect of the study. The boys were particularly drawn to the strategic elements of the game and expressed enjoyment in using competitive strategies, such as Clover Cards or cloud jumps, to win their opponents. Girls also enjoyed the strategic aspects of the game but were more sensitive than boys to the social dynamics, such as team interactions and the challenges of working with peers. Additionally, the differing patterns between boys and girls in interest levels over the three-day gameplay sessions reflect distinct motivational dynamics. These differences suggest the significance of gender-specific factors in game design and educational applications. Understanding how boys and girls engage with the game differently can lead to more inclusive and effective learning experiences for all students.

Unplugged games can serve as powerful tools for teaching CT by providing hands-on, interactive experiences that demystify abstract concepts and make learning accessible to all students, regardless of their technological resources. However, to maximize the impact of such games, it is essential to design them with intentional learning goals and inclusivity in mind, addressing the gameplay mechanics that attract the use of targeted skills and appeal to both genders. Future research should continue to explore the nuanced ways in which unplugged game-based learning environments can be optimized to support diverse learning needs and promote CT skills among all students.

References

Abrantes, J. L., Seabra, C., & Lages, L. F. (2007). Pedagogical affect, student interest, and learning performance. *Journal of Business Research, 60*(9), 960-964. https://doi.org/10.1016/j.jbusres.2006.10.026

Admiraal, W., Huizenga, J., Heemskerk, I., Kuiper, E., Volman, M., & Ten Dam, G. (2014). Gender-inclusive game-based learning in secondary education. *International Journal of Inclusive Education, 18*(11), 1208-1218. https://doi.org/10.1080/13603116.2014.885592

Aksit, O., & Wiebe, E. (2019). Exploring Force and Motion Concepts in Middle Grades Using Computational Modeling: a Classroom Intervention Study. *Journal of Science Education and Technology, 29*, 65-82 (2020). https://doi.org/10.1007/s10956-019-09800-z

Alamer, R. A., Al-Doweesh, W. A., Al-Khalifa, H. S., & Al-Razgan, M. S. (2015).

Programming unplugged: Bridging CS unplugged activities gap for learning key programming concepts. In N. Walker (Ed.), *Proceedings of the Fifth International Conference on e-Learning (ICEEE)* (pp. 97-103). IEEE. https://doi.org/10.1109/ECONF.2015.27

Aleksić, V., & Ivanović, M. (2017). Early adolescent gender and multiple intelligences profiles as predictors of digital gameplay preferences. *Croatian Journal of Education: hrvatski časopis za odgoj i obrazovanje, 19*(3), 697-727. https://doi.org/10.15516/cje.v19i3.2262

Althouse, A. D. (2016). Adjust for Multiple Comparisons? It’s Not That Simple. *The Annals of Thoracic Surgery, 101*(5), 1644-1645.

Ballard, E. D., & Haroldson, R. (2021). Analysis of computational thinking in Children’s literature for K-6 students: Literature as a non-programming unplugged resource. *Journal of Educational Computing Research, 59*(8). https://doi.org/10.1177/07356331211004048

Barker, L. J., & Aspray, W. (2006). The state of research on girls and IT. In J. Cohoon, & W.

Aspray (Eds.), *Women and Information Technology: Research on Underrepresentation.* The MIT Press. https://doi.org/10.7551/mitpress/7272.003.0003

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Acm Inroads, 2*(1), 48-54. https://doi.org/10.1145/1929887.1929905

Barsalou, L. W., & Wiemer-Hastings, K. (2005). Situating abstract concepts. In D. Pecher, & R. A. Zwaan (Eds.), *Grounding cognition: The role of perception and action in memory, language, and thinking* (pp. 129-163). Cambridge University Press. https://doi.org/10.1017/CBO9780511499968.007

Bell, T., & Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work? In H. Böckenhauer, D. Komm, & W. Unger (Eds.), *Adventures Between Lower Bounds and Higher Altitudes* (pp. 497–521). Springer, Cham. https://doi.org/10.1007/978-3-319-98355-4_29

Bilgic, K., & Dogusoy, B. (2023). Exploring secondary school students' computational thinking experiences enriched with block-based programming activities: An action research. *Education and Information Technologies, 28*(28), 10359-10384 (2023). https://doi.org/10.1007/s10639-023-11583-1

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education: Principles, Policy & Practice, 5*(1), 7-74. https://doi.org/10.1080/0969595980050102

Boman, B. (2023). The influence of SES, cognitive, and non-cognitive abilities on grades: cross-sectional and longitudinal evidence from two Swedish cohorts. *European Journal of Psychology of Education, 38*(2), 587-603. https://doi.org/10.1007/s10212-022-00626-9

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in*

Psychology, 3(2), 77-101.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association (AERA).* Vancouver, Canada.

Chang, C. L., & Chen, C. N. (2023). Practical impact of school-age children using digital games to learn environmental education. *E3S Web of Conferences*. https://doi.org/10.1051/e3sconf/202345207005

Chen, P., Yang, D., Metwally, A. H., Lavonen, J., & Wang, X. (2023). The influence of unplugged activities on students' computational thinking skills: a systematic literature review and meta-analysis. *International Journal of STEM Education, 10*(47). https://doi.org/10.1186/s40594-023-00434-7

Choi, K. S. (2015). A comparative analysis of different gender pair combinations in pair programming. *Behaviour & Information Technology , 34*(8), 825-837. https://doi.org/10.1080/0144929X.2014.937460

Chongo, S., Osman, K., & Nayan, N. A. (2021). Impact of the plugged-in and unplugged chemistry computational thinking modules on achievement in chemistry. *EURASIA Journal of Mathematics, Science and Technology Education, 17*(4). https://doi.org/10.29333/ejmste/10789

Cleveland, C., Gabrieli, J. D., Scherer, E., & West, M. R. (2022). Measuring Cognitive Skills in School Settings: Evidence from the NeuroCognitive Performance Test1. Retrieved from https://scholar.harvard.edu/files/chcleveland/files/measuring_cognitive_skills_in_school_settings.pdf

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed* *methods approaches*. Sage publications.

Creswell, J. W., & Plano Clark, V. L. (2017). *Designing and Conducting Mixed Methods Research.* Thousand Oaks, CA: Sage.

Crowe, M. (2003). Jump for the sun II: Can a monthly program change girls' and women's attitudes about STEM? *Journal of Women and Minorities in Science and Engineering, 9*(3-4), 8.

Ezeamuzie, N. O., & Leung, J. S. (2022). Computational thinking through an empirical lens: A systematic review of literature. *Journal of Educational Computing Research, 60(2)*, 481-511. https://doi.org/07356331211033158

Flewitt, R. S. (2013). Interviews. In A. Clark, R. Flewitt, M. Hammersley, & M. Robb (Eds.), *Understanding research with children and young people* (pp. 34-50). London: Sage.

Gibbons, J. D., & Chakraborti, S. (2014). *Nonparametric statistical inference: revised and expanded.* CRC Press.

Gibson, B., & Bell, T. (2013). Evaluation of games for teaching computer science. *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, (pp. 51–60). Aarhus, Denmark. https://doi.org/10.1145/2532748.2532751

Gresse von Wangenheim, C., Araújo e Silva de Medeiros, G., Missfeldt Filho, R., Petri, G., da Cruz Pinheiro, F., Ferreira, M. N., & Hauck, J. C. (2019). SplashCode - A Board Game for Learning an Understanding of Algorithms in Middle School. *Informatics in Education, 18*(2), 259-280. https://doi.org/10.15388/infedu.2019.12

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Hamlen, K. R. (2011). Children's choices and strategies in video games. *Computers in Human*

Behavior, 27(1), 532-539. https://doi.org/10.1016/j.chb.2010.10.001

Harackiewicz, J. M., Smith, J. L., & Priniski, S. J. (2016). Interest Matters: The Importance of Promoting Interest in Education. *Policy insights from the behavioral and brain sciences, 3*(2), 220-227. https://doi.org/10.1177/2372732216655542

Hartl, A. C., DeLay, D., Laursen, B., Denner, J., Werner, L., Campe, S., & Ortiz, E. (2015). Dyadic instruction for middle school students: Liking promotes learning. *Learning and Individual Differences, 44*, 33-39. https://doi.org/10.1016/j.lindif.2015.11.002

Hartmann, T., & Klimmt, C. (2006). Gender and computer games: Exploring females' dislikes. *Journal of computer-mediated communication, 11*(4), 910-931. https://doi.org/10.1111/j.1083-6101.2006.00301.x

Hattie, J., & Timperley, H. (2007). The Power of Feedback. *Review of Educational Research, 77*(1). https://doi.org/10.3102/003465430298487

Homer, B. D., Hayward, E. O., Frye, J., & Plass, J. L. (2012). Gender and player characteristics in video game play of preadolescents. *Computers in Human Behavior, 28*(5), 1782-1789. https://doi.org/10.1016/j.chb.2012.04.018

Jaccheri, L., Pereira, C., & Fast, S. (2020). Gender issues in computer science: Lessons learnt and reflections for the future. *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (pp. 9-16). Timisoara, Romania: IEEE. https://doi.org/10.1109/SYNASC51798.2020.00014

Jin, H., & Cutumisu, M. (2024). Cognitive, interpersonal, and intrapersonal deeper learning domains: A systematic review of computational thinking. *Education and Information Technologies*. https://doi.org/10.1007/s10639-024-12744-6

Kafai, Y. B., & Burke, Q. (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

Kallia, M., & Sentance, S. (2018). Are Boys More Confident Than Girls? The Role of Calibration and Students' Self-Efficacy in Programming Tasks and Computer Science. *13th Workshop in Primary and Secondary Computing Education (WiPSCE '18).*Potsdam, Germany. https://doi.org/10.1145/3265757.3265773

Kinzie, M. B., & Joseph, D. R. (2008). Gender differences in game activity preferences of middle school children: implications for educational game design. *Educational Technology Research and Development, 56*, 643-663. characteristics in video game play of preadolescents. *Computers in Human Behavior, 28*(5), 1782-1789. https://doi.org/10.1016/j.chb.2012.04.018

Kong, S. C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education, 127*, 178-189. https://doi.org/10.1016/j.compedu.2018.08.026

Krippendorff, K. (2013). *Content Analysis: An Introduction to Its Methodology.* Thousand Oaks, CA: Sage Publications.

Küçükkara, M. F., & Aksüt, P. (2021). An Example of Unplugged Coding Education in Preschool Period: Activity-Based Algorithm for Problem Solving Skills. *Journal of Inquiry Based Activities, 11*(2), 81-91. Retrieved from https://www.ated.info.tr/ojs- 3.2.1-3/index.php/ated

Kuo, W. C., & Hsu, T. C. (2020). Learning computational thinking without a computer: How computational participation happens in a computational thinking board game. *The Asia-*

Pacific Education Researcher, 29(1), 67-83. https://doi.org/10.1007/s40299-019-00479-9

Kuo, Y. T., & Kuo, Y. C. (2023). African American Students' Academic and Web Programming Self-Efficacy, Learning Performance, and Perceptions towards Computer Programming in Web Design Courses. *Education Sciences, 13*(12). https://doi.org/10.3390/educsci13121236

Lange, B. P., Wühr, P., & Schwarz, S. (2021). Of Time Gals and Mega Men: Empirical findings on gender differences in digital game genre preferences and the accuracy of respective gender stereotypes. *Frontiers in Psychology, 12*. https://doi.org/10.3389/fpsyg.2021.657430

Lathifah, A., Asrowi, A., & Efendi, A. (2023). Students' Perspectives on Game-Based Learning and Computational Thinking. *International Journal of Information and Education Technology, 13*(3), 597-603. https://doi.org/10.18178/ijiet.2023.13.3.1843

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. SAGE Publications.

Lindberg, R. S., Laine, T. H., & Haaranen, L. (2019). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games. *British journal of educational technology, 50(4)*, 1979-1995. https://doi.org/10.1111/bjet.12685

Liu, H., Chen, H., Yu, S., & Shih, Y. (2022). Effect of Learning Computational Thinking Using Board Games in Different Learning Styles on Programming Learning. In Y. Huang, S. Cheng, J. Barroso, & F. Sandnes (Ed.), *Innovative Technologies and Learning. ICITL 2022* (pp. 514-521). Springer, Cham. https://doi.org/10.1007/978-3-031-15273-3_56

López-Fernández, F. J., Mezquita, L., Griffiths, M. D., Ortet, G., & Ibáñez, M. I. (2021). The role of personality on disordered gaming and game genre preferences in adolescence: gender differences and person-environment transactions. *Adicciones, 33*(3), 263-272. https://doi.org/10.20882/adicciones.1370

Lye, S. Y., & Koh, J. H. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. https://doi.org/j.chb.2014.09.012

Ma, H., Zhao, M., Wang, H., Wan, X., Cavanaugh, T. W., & Liu, J. (2021). Promoting pupils' computational thinking skills and self-efficacy: A problem-solving instructional approach. *Educational Technology Research and Development, 69*(3), 1599-1616. https://doi.org/10.1007/s11423-021-10016-5

Marjanen, P., Mönkkönen, I., & Vanhala, M. (2011). Peer Group Learning During the Board Game Sessions. *5th European Conference on Games Based Learning* (pp. 388-394). The National and Kapodistrian University of Athens.

Mason, S. L., & Rich, P. J. (2020). Development and analysis of the elementary student coding attitudes survey. *Computers & Education, 153*, 153. https://doi.org/10.1016/j.compedu.2020.103898

Master, A., Meltzoff, A. N., & Cheryan, S. (2021). Gender stereotypes about interests start early and cause gender disparities in computer science and engineering. *Psychological and Cognitive Science, 118*(48). https://doi.org/10.1073/pnas.2100030118

Merino-Armero, J. M., González-Calero, J. A., Cózar-Gutiérrez, R., & del Olmo-Muñoz, J. (2022). Unplugged activities in cross-curricular teaching: Effect on sixth graders' computational thinking and learning outcomes. *Multimodal Technologies and Interaction, 6(2)*(13). https://doi.org/10.3390/mti6020013

Monga, M., Lodi, M., Malchiodi, D., Morpurgo, A., & Spieler, B. (2018). Learning to program in a constructionist way. *Proceedings of Constructionism 2018.* Vilnius, Lithuania. Retrieved from https://inria.hal.science/hal-01913065v1

Moors, L., Luxton-Reilly, A., & Denny, P. (2018). Transitioning from Block-Based to Text- Based Programming Languages. *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (pp. 57-64). Auckland, New Zealand: IEEE. https://doi.org/10.1109/LaTICE.2018.000-5

Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the K-12 curriculum? *Journal of Information Technology Education, 15*, 283-303. https://doi.org/10.28945/3521

Nambiar, R. (2020). Coding as an Essential Skill in the Twenty-First Century. In B. Panth, & R. Maclean (Eds.), *Anticipating and Preparing for Emerging Skills and Jobs* (pp. 237- 243). Springer, Singapore. https://doi.org/10.1007/978-981-15-7018-6_29

Nguyen, H. A., Else-Quest, N., Richey, J. E., Hammer, J., Di, S., & McLaren, B. M. (2023). Gender Differences in Learning Game Preferences: Results Using a Multi- dimensional Gender Framework. In N. Wang, G. Rebolledo-Mendez, N. Matsuda, O. Santos, & V. Dimitrova (Ed.), *Artificial Intelligence in Education. AIED 2023. 13916.* Springer, Cham. https://doi.org/10.1007/978-3-031-36272-9_45

Niederle, M., & Vesterlund, L. (2010). Explaining the Gender Gap in Math Test Scores: The Role of Competition. *Journal of Economic Perspectives, 24*(2), 129-144. https://doi.org/10.1257/jep.24.2.129

Nunnally, J. C., & Bernstein, I. H. (1994). *Psychometric Theory.* McGraw-Hill Companies.

Patton, M. Q. (1999). Enhancing the quality and credibility of qualitative analysis. *Health Services Research*, 34(5 Pt 2), 1189–1208.

Polat, E., & Yilmaz, R. M. (2022). Unplugged versus plugged-in: Examining basic programming achievement and computational thinking of 6th-grade students. *Education and Information Technologies, 27*, 9145-9179. https://doi.org/10.1007/s10639-022-10992-y

Poole, F. J., Clarke-Midura, J., Rasmussen, M., Shehzad, U., & Lee, V. R. (2022). Tabletop games designed to promote computational thinking. *Computer Science Education, 32*(4), 449-475. https://doi.org/10.1080/08993408.2021.1947642

Przybylski, A. K., Rigby, C. S., & Ryan, R. M. (2010). A motivational model of video game engagement. *Review of general psychology, 14*(2), 154-166. https://doi.org/10.1037/a0019440

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Rist, S., Chiddambaranathan, M., Escobar, C., & Wiesmann, U. (2006). "It was Hard to Come to Mutual Understanding …"—The Multidimensionality of Social Learning Processes Concerned with Sustainable Natural Resource Use in India, Africa and Latin America. *Systemic Practice and Action Research, 19*, 219-237. https://doi.org/10.1007/s11213-006-9014-8

Riverside Insights. (n.d.). The Iowa Assessments. Retrieved September 13, 2024, from https://riversideinsights.com/the_iowa_assessments

Rothman, K. J. (1990). No Adjustments Are Needed for Multiple Comparisons. *Epidemiology, 1*(1), 43-46.

Ryan, R. M., Rigby, C. S., & Przybylski, A. (2006). The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion, 30*, 344-360.

https://doi.org/10.1007/s11031-006-9051-8

Scirea, M., & Valente, A. (2020). Boardgames and computational thinking: How to identify games with potential to support CT in the classroom. *Proceedings of the 15th International Conference on the Foundations of Digital Games.* Bugibba, Malta. https://doi.org/10.1145/3402942.3409616

Selby, C. C. (2012). Promoting computational thinking with programming. *Proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE '12)* (pp. 74-77). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2481449.2481466

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

Srisupawong, Y., Koul, R., Neanchaleay, J., Murphy, E., & Francois, E. J. (2018). The relationship between sources of self-efficacy in classroom environments and the strength of computer self-efficacy beliefs. *Education and Information Technologies, 23*, 681-703. https://doi.org/10.1007/s10639-017-9630-1

Sun, D., Ouyang, F., Li, Y., & Zhu, C. (2021). Comparing learners' knowledge, behaviors, and attitudes between two instructional modes of computer programming in secondary education. *International Journal of STEM Education, 8*, 1-15. https://doi.org/10.1186/s40594-021-00311-1

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*. https://doi.org/10.1016/j.compedu.2019.103798.

Tsarava, K., Moeller, K., & Ninaus, M. (2018). Training computational thinking through board games: The case of Crabs & Turtles. *International Journal of Serious Games, 5*(2), 25-44. https://doi.org/10.17083/ijsg.v5i2.248

Tuğtekin, U., Tuğtekin, E., & Dursun, Ö. (2018). Analysis of readiness for change and self-efficacy perceptions of it stakeholders. *Mersin Üniversitesi Eğitim Fakültesi Dergisi, 14*(3), 1200-1221. https://doi.org/10.17860/mersinefd.354881

Turchi, T., Fogli, D., & Malizia, A. (2019). Fostering computational thinking through collaborative game-based learning. *Multimedia Tools and Applications, 78*(10), 13649-13673. https://doi.org/10.1007/s11042-019-7229-9

Videnovik, M., Vold, T., Kiønig, L., Bogdanova, A. M., & Trajkovik, V. (2023). Game-based learning in computer science education: a scoping literature review. *International Journal of STEM Education, 10*. https://doi.org/10.1186/s40594-023- 00447-2

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49(3)*, 33-35. https://doi.org/1118178.1118215

Witherspoon, E. B., Schunn, C. D., Higashi, R. M., & Baehr, E. C. (2016). Gender, interest, and prior experience shape opportunities to learn programming in robotics competitions. *International Journal of STEM Education, 3*(18). https://doi.org/10.1186/s40594-016-0052-1

Wu, L., Looi, C. K., Multisilta, J., How, M. L., Choi, H., Hsu, T. C., & Tuomi, P. (2020). Teacher's perceptions and readiness to teach coding skills: a comparative study between Finland, Mainland China, Singapore, Taiwan, and South Korea. *The Asia- Pacific Education Researcher, 29*(1), 21-34. https://doi.org/10.1007/s40299-019-00485-x

Wu, S. (2018). The Development and Challenges of Computational Thinking Board Games. *1st International Cognitive Cities Conference (ICS3)*, (pp. 129-131). Okinawa, Japan.

https://doi.org/10.1109/IC3.2018.00-45

Yang, D., & Kopcha, T. J. (2022). Designing a Board Game for Beginning Block-Based Programmers. *International Journal of Designs for Learning, 13*(1), 35-45. https://doi.org/10.14434/ijdl.v13i1.32211

Yin, R. K. (2018). *Case Study Research and Applications: Design and Methods.* Thousand Oaks, CA: Sage.

Yoon, C. S., & Khambari, M. N. (2022). Design, Development, and Evaluation of the Robobug Board Game: An Unplugged Approach to Computational Thinking. *International Journal of Interactive Mobile Technologies, 16*(6), 41-60.

Zachopoulou, E., Trevlas, E., & Tsikriki, G. (2004). Perceptions of gender differences in playful behaviour among kindergarten children. *European Early Childhood Education Research Journal, 12*(1), 43-53. https://doi.org/10.1080/13502930485209301

Zallio, M. (2021). Democratizing Information Visualization. A Study to Map the Value of Graphic Design. In M. Soares, E. Rosenzweig, & A. Marcus (Ed.), *Design, User Experience, and Usability: UX Research and Design (HCII 2021)* (pp. 495-508). Springer, Cham. https://doi.org/10.1007/978-3-030-78221-4_33

Zdawczyk, C., & Varma, K. (2022). Engaging girls in computer science: gender differences in attitudes and beliefs about learning scratch and python. *Computer Science Education, 33*(4), 600-620. https://doi.org/10.1080/08993408.2022.2095593

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*. https://doi.org/10.1016/j.compedu.2019.103607

Zhang, Y., Liang, Y., Tian, X., & Yu, X. (2024). The effects of unplugged programming activities on K-9 students' computational thinking: meta-analysis. *Educational technology research and development, 72*, 1331-1356. https://doi.org/10.1007/s11423-023-10339-5

Zhong, B., Wang, Q., & Chen, J. (2016). The impact of social factors on pair programming in a primary school. *Computers in Human Behavior, 64*, 423-431. https://doi.org/10.1016/j.chb.2016.07.017

APPENDICES

Appendix B

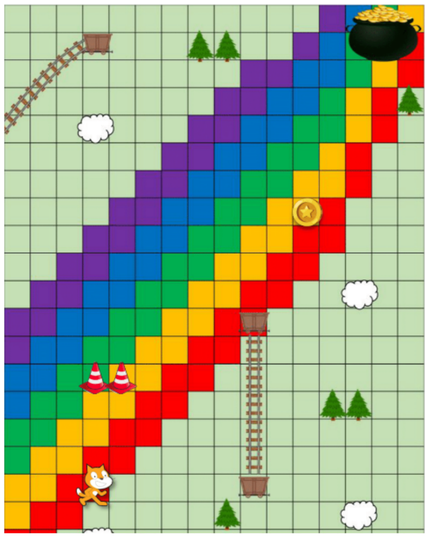
Pre- and Posttest

Your Name: ____________

- Have you had any experience with coding? YES NO
- If you answered yes, which programming language did you use?
  (Example: Scratch, Python, Java, Blockly, etc.)

  ____

The cat wants to reach the gold pot while taking the coin along the way. List the necessary steps using the provided code blocks. (You can use a code block multiple times. You do not have to use all of them.)



Move __ steps forward

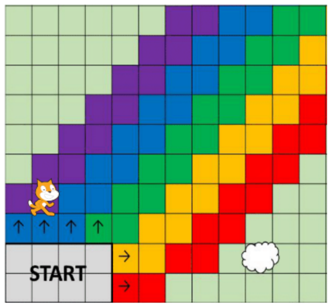
Collect coin

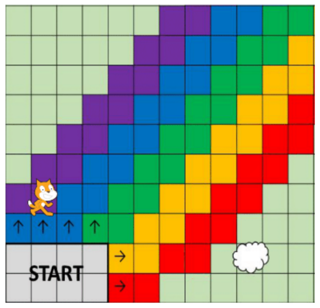
Turn [↺/↻] 90 degrees

Get on the wagon (Get off at the other end)

Your Answer:

Circle the spot where the cat will arrive when following the code below.





Repeat below 2 times

Move 3 steps forward

If on yellow, then | Turn ↺ 90 degrees | Move **5** steps forward

Repeat below 2 times

Move 3 steps forward

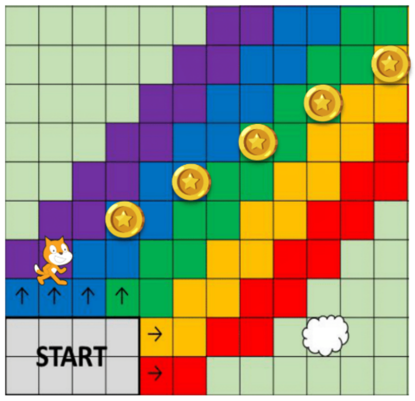
If on yellow, then | Turn ↺ 90 degrees | Move **3** steps forward

Then,

Move 2 steps forward

If on blue, then | Move 1 step forward

Have the cat collect all five coins using as **less number of** code blocks as possible. (You can use a block multiple times. You do not have to use all of them.)



Move __ steps forward

Collect coin

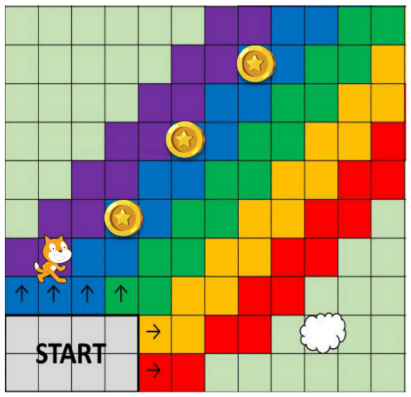
Turn [↺/↻] 90 degrees

If on [color], then

Repeat below ___ times

Your Answer:

Have the cat collect all three coins using as **less number of** code blocks as possible. (You can use a block multiple times. You do not have to use all of them.)



Move __ steps forward

Collect coin

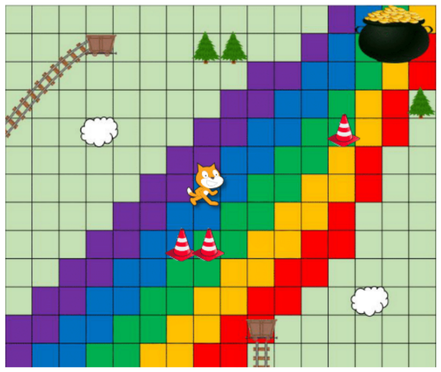
Turn [↺/↻] 90 degrees

If on [color], then

Repeat below ___ times

Your Answer:

The cat wants to reach the gold pot, but there was ONE error in the code. Find the code block with the error and fix it so the cat can reach the pot.



Repeat below [2] times

Move [3] steps forward

If on [red], then Turn [↺] 90 degrees

Your Answer:

3

Note. Scratch is a project of the Scratch Foundation, in collaboration with the Lifelong Kindergarten Group at the MIT Media Lab. It is available for free at https://scratch.mit.edu.

Appendix C

Scoring Rubric for Pre- and Posttest

Points	**Description**	**Criteria**
4	Excellent / Ideal Solution	The task is completed with no errors. The solution is clear, well-organized, and demonstrates full understanding.
3	Good / Functional but Not Ideal	The task is completed with 1-2 minor errors or could be improved in terms of clarity, efficiency, or organization.
2	Satisfactory / Partial Success	The task shows progress toward completion but contains 3-4 notable errors or omissions that impact effectiveness.
1	Needs Improvement / Major Gaps	The task demonstrates limited understanding, with 5 or more significant errors or incomplete implementation.
0	No Response / Off-task	No response provided, or the submission is entirely unrelated to the task.

Appendix D

Interest Survey

GAMEPLAY EXPERIENCE

1. What is your name? ___________________
2. What grade are you in? ________________
3. How much did you enjoy the game so far? (Circle the face that shows how you felt.)

😟 🙁 😕 😐 🙂 😀 😍

4. What did you enjoy the **most** about the game so far?

5. What did you enjoy the **least** about the game so far?

6. Compared to the last time I played the game, today, I felt like the game was:

   ☐ More interesting to play

   ☐ Similarly interesting to last time

   ☐ Less interesting to play

   Please explain why you felt that way:

CODING INTEREST

For each of the following statements, choose one answer that shows how much you agree.

	STRONGLY DISAGREE	DISAGREE	SOMEWHAT AGREE	AGREE	STRONGLY AGREE
I like coding, or I think I would like coding.					
I would like to learn more about coding.					
Solving coding problems seems fun.					
Coding is interesting					
I would like to study coding in the future					

CHAPTER 5

CONCLUSION

This dissertation explores the design, implementation, and impact of a board game developed to enhance computational thinking (CT) skills in K-12 classrooms, with the aim to create a practical tool for classroom integration while advancing the theoretical understanding of game-based CT learning. The project addresses key challenges in CT education, particularly for younger students who struggle with abstract coding concepts and limited technology access (Grover & Pea, 2013; Lye & Koh, 2014). Computer-based methods can be intimidating for beginners and require extensive scaffolding (Kafai & Burke, 2015). Unplugged activities, such as board games, offer a tangible, hands-on alternative, making CT concepts more accessible and engaging (Bell & Vahrenhold, 2018). They also alleviate issues related to technology access, making them suitable for diverse educational contexts (Bell & Vahrenhold, 2018).

By providing an unplugged, hands-on approach, this body of work contributes to both practical and theoretical knowledge in CT education, supporting the use of alternative methods to engage a broader range of learners (Shute et al., 2017; Yadav et al., 2014). The iterative development and classroom testing of the board game *Lucky Codes* demonstrate such games' potential to enhance CT learning in a variety of settings, offering valuable insights for educators and game developers.

Summary of Findings

The first study focused on the initial design and development of the board game, with the goal of creating a tool that could effectively teach fundamental CT concepts such as sequencing, loops, and conditionals in an unplugged format (Yang & Kopcha, 2022). The study highlighted the challenges in creating a game that is both educational and engaging, emphasizing the need for continuous and intentional refinement of game mechanics. Key insights included the importance of clearly defined goals for guiding design decisions, the use of immediate feedback and exclusive rewards to engage learners in more advanced concepts, and the benefit of low-fidelity prototypes for gathering honest feedback. This study provided insights into the design process of an educational game and set the foundation for subsequent studies by establishing a functional and theoretically grounded game prototype.

Building on the insights gained from the initial study, the second study refined and implemented the final version of the game in K-12 classroom settings through a pilot study with four students. This phase aimed to provide an initial assessment of the game's effectiveness in promoting CT skills. The findings demonstrated that the game effectively facilitated the use of core CT concepts, including sequencing, decomposition, conditionals, and loops. By the end of the intervention, students showed increased complexity and efficiency in applying these skills, highlighting the potential of such hands-on, unplugged tools to make abstract CT concepts more accessible to young learners. This study provided an initial understanding of the game's educational impact and also informed the research questions for the subsequent study, guiding further exploration of its educational potential.

The final study expanded upon the findings of the previous phases by conducting a comprehensive evaluation of the game's impact on student learning and engagement. This phase explored not only how students' CT skills developed over time but also how the game influenced their attitudes toward programming and problem-solving with the additional examination of gender differences. The results showed that sustained use of the board game led to continued improvement in CT skills, particularly in the enhanced efficiency in the use of conditionals and loops. Additionally, the study noted fluctuations in student interest and engagement by gender, suggesting the need for ongoing adjustments to maintain motivation and learning efficacy. This study validated the effectiveness of the board game as a learning tool for CT and provided insights into the sustainability of engagement. It highlighted the potential of unplugged games to actively promote CT skill development while making the learning process appealing and accessible to both genders.

Each of these studies informed and built upon the previous one. The testing of initial prototypes guided the design refinements for the classroom implementation, while the classroom findings informed the comprehensive evaluation of the game's effectiveness in learning and engagement. This progression ensured that the board game evolved to meet the practical needs of educators and the educational requirements of students, ultimately contributing both to the development of a viable educational tool and advancing the theoretical understanding of how unplugged games can support CT learning in K-12 education.

Implications

For Educators and Teachers

Educators can utilize board games like *Lucky Codes* as a practical tool to supplement CT instruction. The findings show that the game effectively supports the learning of complex CT concepts, such as loops and conditionals, through a tangible, hands-on approach. Teachers can incorporate the game into their lesson plans to provide an interactive and engaging way to reinforce these abstract concepts, making CT more accessible to students who may struggle with purely digital or text-based programming exercises. However, it is important to note that students' overall interest in playing the game tended to decrease by the third session. To maintain interest, educators could introduce additional challenges or obstacles as students become more proficient with the game, ensuring the activity remains stimulating and challenging.

The dissertation also demonstrated that the game was appealing and effective for both boys and girls, helping to bridge the gender gap often seen in programming-related subjects. When implementing such games in the classroom, teachers should consider the varying levels of interest and preferences between boys and girls to ensure that all students are equally engaged. For example, while both boys and girls enjoyed strategizing, boys tended to be more competitive, whereas girls were more sensitive to the social aspects of gameplay. To address these differences, teachers might group students by gender or emphasize collaborative strategies and set rules to prevent excessive competition, creating an inclusive and supportive learning environment for all students.

For Educational Game Developers

The dissertation emphasized that having clearly defined educational goals is essential for guiding game design decisions, highlighting the need for developers to establish specific learning objectives early in the development process. This ensures that all game mechanics, narratives, and challenges are purposefully aligned with these goals, creating an educational experience that is both engaging and effective. Integrating features such as immediate feedback and exclusive rewards can further enhance this by maintaining student engagement and encouraging the exploration of advanced concepts. The findings suggest that these elements improve the educational value of games, making abstract CT concepts more approachable and motivating students to persist through challenges. Developers should also adopt an iterative approach by using low-fidelity prototypes to gather feedback during the design process, refining their games based on user input. This method helps ensure that the final product is well-tuned to the educational needs of its audience, effectively supporting learning outcomes and enhancing the overall learning experience.

By synthesizing these findings, educators and game developers can create more effective learning environments that actively engage diverse students in developing computational thinking skills. Whether through adapting gameplay to maintain student interest or fine-tuning design to align with specific educational goals, the implications of this study highlight the potential for board games like *Lucky Codes* to transform how CT is taught and learned. This work opens new avenues for future research and innovation in unplugged learning tools, offering practical and theoretical contributions to the field of computational thinking education.

References

Bell, T., & Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work? In H. Böckenhauer, D. Komm, & W. Unger (Eds.), *Adventures Between Lower Bounds and Higher Altitudes* (pp. 497–521). Springer, Cham. https://doi.org/10.1007/978-3-319-98355-4_29

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Kafai, Y. B., & Burke, Q. (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

Lye, S. Y., & Koh, J. H. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. https://doi.org/j.chb.2014.09.012

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

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), 1-16. https://doi.org/10.1145/2576872

Yang, D., & Kopcha, T. J. (2022). Designing a Board Game for Beginning Block-Based Programmers. *International Journal of Designs for Learning, 13*(1), 35-45. https://doi.org/10.14434/ijdl.v13i1.32211