Promoting Computational Thinking Through Programming in Early Childhood Education: A Mixed-Methods Study in Chinese Kindergartens

by

ZENG, Yue

A Thesis Submitted to

The Education University of Hong Kong

in Partial Fulfillment of the Requirement for

the Degree of Doctor of Education

December 2023

Statement of Originality

I, ZENG, Yue, hereby declare that I am the sole author of the thesis and the material presented in this thesis is my original work except those indicated in the acknowledgement. I further declare that I have followed the University's policies and regulations on Academic Honesty, Copyright and Plagiarism in writing the thesis and no material in this thesis has been submitted for a degree in this or other universities.

Abstract

Computational thinking (CT), a critical competency everyone should possess in the digital age, is attracting increasing attention from researchers and educators worldwide. Programming education, the most crucial way to foster CT, is being introduced into early childhood education (ECE) settings. However, early programming and CT education are still in their infancy, with many unresolved issues. The inconsistency in determining "what to teach" in early programming and CT curricula stands as a primary concern, alongside unexplored challenges in "how to teach, " including the pedagogical issues and selection of suitable programming tools for young children. To address these gaps, Study 1 conducted a systematic review to propose a CT curriculum framework for ECE that addresses "what to teach". This framework clarifies the content that should be included in the early CT curriculum, serving as a foundation for developing early childhood CT education. Study 2 and Study 3 focused on addressing the issue of "how to teach." Building on the CT curriculum framework for ECE, study 2 delved into a case study of an early childhood teacher to examine her content knowledge (CK) and pedagogical knowledge (PK) in early programming and CT education through the analysis of "what was taught" and "how CT was taught." By identifying knowledge gaps, misconceptions, and teaching challenges among teachers, this study offers insights for improving professional knowledge and teaching effectiveness in this burgeoning field. Additionally, Study 3 focused on another critical aspect

of programming and CT education, i.e., programming tools. This study validated the positive impact of a particular kit in fostering CT skills among young children, offering valuable insights for educators in selecting appropriate programming tools. In conclusion, by proposing a CT curriculum framework for ECE, exploring the CK and PK of an ECT, and investigating the effectiveness of a hybrid kit, this thesis advances our understanding of "what to teach" and "how to teach" programming and CT in the early years.

Keywords: computational thinking, programming, early childhood education

Acknowledgments

It was not until my supervisors kindly reminded me that I had fulfilled the graduation requirements for I have published three SSCI papers that I realized my doctoral study was about to be completed. Although only three in number, these papers have given me the confidence to enter the realm of academic research and the determination to pursue an academic research career.

These achievements could not have been achieved without the invaluable guidance of my two dear supervisors, Dr Weipeng Yang and Professor Alfredo Bautista, from The Education University of Hong Kong. Both supervisors have made remarkable achievements in their respective research fields and demonstrated exceptional dedication to their students. However, they also possess unique qualities when it comes to mentoring students. Dr Yang stands out for his promptness. Whenever I submitted a paper to him, he would always provide feedback within a maximum of two days. What's more, his expertise extended beyond shaping the paper's overall framework; he would delve into the realm of formatting, word choice, and countless other intricate details. This timely response gave me the invaluable opportunity to compare my thinking with Dr Yang's. As a result, I was able to gradually absorb his distinct perspective and method for thinking and conducting research. Despite maintaining a high standard of academic rigor, Dr. Yang was incredibly accommodating and trusting in other aspects, always respecting my decisions. It was within this relaxed and supportive learning environment that I completed my doctoral studies.

On the other hand, Professor Alfredo is known for his meticulousness. Every time I receive his feedback, I am greeted with a sea of revisions and annotations. He diligently guides me step by step, showing me exactly where and how to make improvements, while also explaining the reasoning behind those revisions. Through this meticulous approach, I have gained a profound understanding of effective academic writing methods and techniques,

empowering me to apply these skills flexibly in the future. Moreover, Professor Alfredo is an incredibly encouraging supervisor. He possesses a unique ability to uncover the shining moments in my work and offers genuine affirmation, boosting my confidence in academic research with each interaction.

As a working mother pursuing a doctoral degree, I am deeply grateful for the silent support of my family members. Whether I was pursuing a doctoral degree or not, my life has always been filled with a busy work and study schedule. Therefore, I hold an immense appreciation for the continuous support of my spouse, and I firmly believe that his unwavering support will continue in the future. I would also like to express my special thanks to my two wonderful sons, who brought me immeasurable joy and never burdened me. They are the angels who have graced my home and are the source of my happiness.

Furthermore, I consider myself incredibly fortunate to have crossed paths with three extraordinary friends during my doctoral journey - Wang Chan, Xu Rongrong, and Qiu Shiqi. Dr. Wang Chan possesses a solid academic foundation. Whenever I encounter difficulties, I frequently turn to her for advice, and her unwavering patience and meticulousness resemble that of an exceptional "teacher." As for my classmates, Xu Rongrong and Qiu Shiqi, we have shared the same worries and faced similar challenges. We never hesitate to lend a helping hand to one another, for we understand the importance of mutual support and encouragement. Therefore, pursuing an EdD did not challenge me; instead, it made me feel incredibly warm-hearted.

I would also like to thank Wenzhou University, a great university where I work, for its support and assistance throughout my doctoral journey, especially in accommodating my work arrangements.

At this very moment, I feel like a sailor setting sail on the vast and exhilarating academic ocean. I hold a deep hope that even beyond graduation, I can still seek guidance from my

esteemed advisors whenever I encounter challenges. And without a doubt, I know they will be there for me. I am incredibly fortunate to have crossed paths with such exceptional mentors on my academic journey. With their unwavering support, I am determined to continue learning, exploring, and advancing with boundless enthusiasm and an unwavering commitment to excellence on the exciting path that lies ahead.

Table of Contents

List of Abbreviations

CT	Computational Thinking
ECE	Early Childhood Education
ECTs	Early Childhood Teachers
PCK	Pedagogical Content Knowledge
CK	Content Knowledge
PK	Pedagogical Knowledge
GS	Graduate School
EdUHK	The Education University of Hong Kong

List of Figures

List of Tables

Preface

Chapter 2 is extracted from a published article entitled 'Computational Thinking in Early Childhood Education: Reviewing the Literature and Redeveloping the Three-Dimensional Framework.' The work is cited as:

Zeng, Y., Yang, W., & Bautista, A. (2023). Computational thinking in early childhood education: Reviewing the literature and redeveloping the three-dimensional framework. *Educational Research Review*, *39*, 100520. https://doi.org/10.1016/j.edurev.2023.100520

Chapter 3 is extracted from a published article entitled 'Teaching Programming and Computational Thinking in Early Childhood Education: A Case Study of Content Knowledge and Pedagogical Knowledge.' The work is cited as:

Zeng, Y., Yang, W., & Bautista, A. (2023). Teaching programming and computational thinking in early childhood education: a case study of content knowledge and pedagogical knowledge. *Frontiers in Psychology*, *14*. https://doi.org/10.3389/fpsyg.2023.1252718

Chapter 4 is extracted from a published article entitled 'Developing Young Children's Computational Thinking through Programming with a Hybrid Kit.' The work is cited as:

Zeng, Y., Yang, W., & Bautista, A. (accepted). Developing Young Children's Computational Thinking through Programming with a Hybrid Kit. *Journal for the Study of Education and Development*.

Chapter 1: Introduction

1.1 Defining computational thinking and programming

This section defines two key concepts that underpin my study: computational thinking and programming.

Computational thinking (CT) can be traced back to the constructionist endeavors of Seymour Papert and was initially introduced as a term in a seminal article by Wing (2006). Wing elucidated that CT encompasses the capacity to engage in problem-solving, system design, and comprehension of human behavior by leveraging the foundational principles of computer science. In essence, CT embodies the skill to analyze and subsequently resolve various problems. Her assertions introduced a novel viewpoint on the relationship(s) between humans and computers, prompting a surge of scholarly inquiry into CT.

A commonly adopted definition posits that CT delineates the thought processes implicated in problem formulation and the construction and/or deconstruction of the sequential steps of a solution in a format executable by a computer, a human, or a hybrid of both (Aho, 2011; Kim and Lee, 2016; Wing, 2011). In this manner, CT epitomizes a form of analytical thinking that bears resemblance to mathematics thinking (e.g., problem-solving), engineering thinking (design and assessment of processes), and scientific thinking (systematic analysis) (Bers, 2010; Bers, 2021).

In the context of early childhood education, we define CT as thought processes that young children develop through systematic analysis, exploration, and testing of solutions to open-ended and often complex problems (Wang et al., 2020).

Programming refers to developing a set of instructions that a computer can understand and execute and debugging, organizing, and applying that code to appropriate problem-solving contexts (Mills et al., 2021).

CT and programming are closely **interconnected**, with each relying on and enhancing the other. Programming necessitates CT skills to create efficient and effective code (Lye & Koh, 2014), while programming plays a crucial role in the development of CT (Voogt et al., 2015). For example, when programming, a programmer often needs to break down a complex task into smaller parts, recognize patterns in data, and identify the most efficient approach for each step. This process involves CT skills such as pattern recognition, algorithmic thinking, and abstraction, which can then be applied to other domains, such as mathematics, science, and engineering.

1.2 CT and early learning and development

Since CT points to "the systematic analysis, exploration, and testing of solutions to open-ended and often complex problems" (Wang et al., 2020, p. 78), it is critical for children to develop practical planning skills, critical thinking, and problem-solving abilities. According to Bers (2018), CT plays a vital role in equipping children with problem-solving skills and creativity, particularly in an increasingly digital world. Moreover, CT has been found to have a significant impact on children's metacognition (Mills et al., 2021), executive functions (Di Lieto et al., 2017) and self-regulation (Yang et al., 2022). Beyond enabling learners to engage with computers, CT holds wide-ranging implications for children's learning across various subjects, including reading, writing, mathematics, and social-emotional development (Mills et al., 2021; Wing, 2011). Consequently, the cultivation of CT has emerged as a crucial educational objective and is progressively being integrated into the domain of ECE.

1.3 Overview of global programming and CT initiatives in ECE[1]

Programming and CT education are attracting increasing attention from researchers

[1] This section references the following paper: Bers, M. U., Strawhacker, A., & Sullivan, A. (2022). The state of the field of computational thinking in early childhood education. https://doi.org/10.1787/19939019

and educators in ECE around the world and have been progressively incorporated into ECE (Bers et al., 2022). Current nationwide programming and CT initiatives primarily focus on primary and secondary school children; however, an increasing number of countries and regions have adopted explicit policies and strategies for introducing technology and computer programming to young children. This section, categorized by global regions, delineates existing CT initiatives in ECE.

1.3.1 Americas

In the Americas, the United States is prominently advancing the promotion and implementation of CT educational programs, although several other nations are preparing to launch CT education within or beyond the school curriculum. The US-based Code.org initiative has achieved significant success in encouraging school-aged students to learn computer science and coding skills through initiatives like Hour of Code, offering free resources for schools to engage students as young as kindergarten in 1-hour curricular activities and events.

While Canadian provincial and territorial early learning frameworks do not explicitly mention CT, references to related terms and practices such as "technological competence" in kindergarten emphasize the understanding and application of technological tools for problem-solving. National policy frameworks in Canada, such as the "Digital Action Plan for Education and Higher Education" and the "Educating for a Digital World" report, outline strategies for integrating coding and robotics into education from an early age. In the Canadian territory of British Columbia (BC), children aged 5-8 are supported by the BC Early Learning Framework and curriculum, which includes Applied Design, Skills, and Technologies to foster CT skills among children. Furthermore, digital literacy frameworks in Alberta, BC, and the Northwest Territories, starting from kindergarten, underscore the use of technology to foster innovation and exploration in students rather than solely as a teaching

aid for educators.

Other countries in the Americas, including Chile, Argentina, Uruguay, and Brazil, are incorporating computational and digital technology proficiencies into their national curricula, with a specific focus on CT skills such as decomposition, pattern recognition, and abstraction from early childhood.

1.3.2 Europe

In Europe, a diverse array of CT initiatives is underway across multiple countries. A survey across 21 European nations revealed that coding is integrated into the curriculum at a national, regional, or local level in 16 countries, including Austria, Bulgaria, Czech Republic, Denmark, Estonia, France, Hungary, Ireland, Israel, Lithuania, Malta, Spain, Poland, Portugal, Slovakia, and the United Kingdom (England). The United Kingdom's national curriculum framework introduced in 2013 emphasized computing as an educational domain from early childhood.

In Finland, programming is mandated for all primary school students since 2016, while Estonia and Italy are actively integrating programming and computer science into their curricula. Spain has recognized the significance of CT in education, with various national and regional initiatives like the School of Computational Thinking (EPCIA) and the integration of CT content into primary education in Navarra, Madrid, and Catalonia. National curriculum decrees in Spain stress the development of digital competence and CT throughout compulsory education, starting from early childhood. Additionally, Finland and Belgium are incorporating CT skills into their curriculum frameworks to enhance children's information and communication technology competence from early childhood.

1.3.3 Asia, Australia, and Pacific Island nations

In the Asia-Pacific region, countries like Korea, Taiwan, Hong Kong, and China are implementing national curricular reforms to address the growing emphasis on CT education.

Singapore has launched nationwide projects, such as the PlayMaker initiative, to introduce programming and various technologies into early childhood classrooms, while Australia and New Zealand are revising their curricula to include computer science and digital technologies. Australian childcare services are required to base their educational programs on an approved learning framework, enabling educators to cater to individual developmental needs and interests, including engaging with information and communication technologies for information access and idea exploration. An update to these frameworks is underway to ensure alignment with contemporary developments in practice and knowledge.

1.4 Obstacles hinder programming and CT education in ECE in China

In 2022, China issued the Compulsory Education Information Technology Curriculum Standards (《义务教育信息科技课程标准》), which include CT as one of the core literacies of the information technology curriculum (Ministry of Education, 2022). However, this policy primarily targets primary and secondary school students. CT education in ECE is still in its early stages, lacking necessary policy support. The integration of CT education in ECE faces various obstacles.

1.4.1 Teacher preparedness

Many early childhood educators may not have received training in teaching programming or CT education. Without a solid knowledge of programming and CT education, educators may lack confidence in their ability to teach it effectively, which can hinder the implementation of a programming curriculum.

1.4.2 Limited resources

ECE institutions in China may have limited resources, such as access to computers, software, and other educational robots needed for programming education. Without proper resources, it can be challenging to teach programming to young children effectively.

1.4.3 Cultural attitudes towards technology

Some parents and educators in China may hold traditional beliefs about the role of technology in education, viewing it as a distraction or not essential for young children. These cultural attitudes can hinder the effective implementation of programming education in ECE.

1.4.4 A lack of curriculum alignment

The existing ECE curriculum in China may not be aligned with programming education, making it difficult for teachers to incorporate programming into their lesson plans. Without a clear framework for integrating programming education, it can be challenging to teach these skills to young children effectively.

1.4.5 Curriculum overload

The existing curriculum may already be packed with subjects considered fundamental, leaving little room for the addition of programming education.

1.4.6 Policy and government support

There may be limited government policy supporting the integration of programming education in ECE. Without incentives or support from educational authorities, schools may not prioritize adopting programming into their curriculum.

1.4.7 Societal understanding of programming and perception of relevance

Educators and parents may lack understanding about the importance of programming education and its relevance to young children. The long-term benefits of programming education in developing problem-solving skills and CT may not be widely recognized.

1.4.8 Developmental concerns

There may be concerns about whether young children are developmentally ready to engage with abstract concepts involved in programming.

1.4.9 Assessment

There may be a lack of appropriate assessment methods to evaluate young children's

progress in programming education.

To address these obstacles, a multifaceted approach is needed, which includes policy reform, teacher training, curriculum development, investment in resources, and efforts to shift cultural perceptions about the value of programming education in ECE. Collaboration between educators, policymakers, parents, and the wider community is essential to create an environment where programming education can be effectively integrated into early childhood education in China.

1.5 Pedagogical Issues Related to Teaching Programming and CT in ECE

This section summarizes the teaching context, activity structure, pedagogical approaches, and pedagogical strategies previously used to foster children's programming and CT skills.

1.5.1 Teaching Context

Lee and Junoh (2019) noted the importance of infusing programming and CT into children's daily lives and setting up programming centers/corners in early childhood classrooms. Mills et al. (2021) emphasized that integrating programming and CT into other learning domains would provide meaningful learning contexts for young children.

1.5.2 Activity Structure

There are three categories of programming and CT activity structure: *highly structured, mixed, and open-ended*. Most studies designed highly structured programming and CT activities (Khoo, 2020; Nam et al., 2019) and few studies designed open-ended free play with programming tools. Newhouse et al. (2017) found that the children appeared more engaged and motivated in the high teacher-supported sessions rather than in free play without explicit scaffolding. Other studies designed mixed activities (Bers et al., 2014; Bers et al., 2019). For instance, in the study by Strawhacker and Bers (2015), there was always a "buffer lesson" for children to explore the programming materials freely, which allowed them to

absorb what they had learned and kept their attention throughout other highly structured activities.

1.5.3 Pedagogical Approaches

Early programming and CT education employs a variety of pedagogical approaches. One such approach is the task-based approach, where learning activities revolve around tasks guided by adults (McCormick & Hall, 2021). Bers (2019) showed how such intentionally structured activities can aid young children in developing CT skills. Another notable approach is the project-based learning, characterized by its student-centered nature. This approach emphasizes students' autonomy, goal-setting, planning, exploration, cooperation, and reflection within authentic real-world practices (Kokotsaki et al., 2016). Several studies involved activities of the construction of robots, engaging students in design, problem-solving, decision-making, and investigative tasks (Macrides et al., 2021). Play-based learning, on the other hand, presents a playful and child-directed pedagogical approach with some adult guidance and predefined learning objectives (Pyle & Danniels, 2017). Critten et al. (2022) suggested play-based, pedagogic practices can be used with children as young as 2 years to learn many of the basic concepts involved in CT skills. Moreover, Lye and Koh (2014) suggested designing a problem-solving learning environment, which includes authentic problems, information processing, scaffolding and reflection, to enhance students' CT practices and perspectives.

1.5.4 Pedagogical Strategies

Previous studies have examined the effectiveness of different pedagogical strategies for improving young children's CT, including unplugged activities, embodied cognition, external memory support scaffolding, and pair programming. Unplugged programming uses materials like paper, cards, and blocks and has been shown to improve CT skills through embodied learning, lower cognitive load, and concrete analogies (Otterborn et al., 2020;

Romero et al., 2018). While for embodied cognition, there are two kinds of embodiment according to the source of body movement: direct embodiment, which refers to moving bodies to perform solution steps; and surrogate embodiment, which refers to manipulating an external surrogate without engaging their bodies (Fadjo, 2012b). External memory support scaffolding is used to help children cope with working memory limitations and reduce cognitive load during programming (Angeli & Valanides, 2020). Pair programming, a collaborative programming approach in which two students work together on a single computer to complete the same programming task, positively improved students' programming and CT skills, learning motivation, metacognition, and collaboration (Denner et al., 2014; Papadakis, 2018). Besides these experimental studies, Wang et al. (2020) video observed various strategies an exemplary teacher used to support preschoolers' CT skills, such as modelling a positive attitude toward error, breaking down problems into small steps, and providing different scaffolds according to children's individual needs.

However, previous studies were mainly aimed at validating the effectiveness of a particular pedagogical strategy in improving children's CT without examining what pedagogical strategies teachers used. Only Wang et al. (2020) investigated the pedagogical strategies used by a male teacher; however, this case study was conducted in a higher teacher-student ratio (1:3) context instead of a large-group context which is common in Asian cultural contexts.

1.6 Tools for early CT learning

Yu and Roque (2019) classified programming tools into physical, virtual, and hybrid kits. Physical kits consist of tangible components. Virtual kits are PC and/or mobile-device-based applications without tangible components. Hybrid kits combine both tangible and virtual parts and can further be divided into two subcategories: "kits with physical robot and graphical programming environment" and "kits with virtual sprites and tangible programming

environment" (Yu & Roque, 2019, p. 23). Previous studies that investigating the effectiveness of programming on young children's CT have primarily employed physical kits, such as Bee-Bot (Angeli & Valanides, 2020), KIBO (Bers et al., 2019), and Code-a-pillar (Wang et al., 2020). Additionally, some studies have utilized virtual kits, such as ScratchJr (Strawhacker et al., 2018) and Code.Org (Çiftci & Bildiren, 2020). There has also been exploration of hybrid kits combining a physical robot with a graphical programming environment, such as LEGO WeDo (Elkin et al., 2014). However, no studies have yet examined the effectiveness of hybrid kits with virtual sprites and tangible programming environment in promoting CT in young children (Yu & Roque, 2019).

1.7 Research gaps, objectives and questions

Although both research and policies indicate the significance of teaching programming to young children, programming and CT education in early childhood is still in its infancy. Many problems of the teaching of programming and CT in ECE settings necessitates a thorough investigation using rigorous theoretical and methodological approaches (Zapata-C et al., 2021).

First and foremost, there is inconsistency across early childhood CT curricula regarding the content to be taught, which holds significance across all disciplines (So et al., 2020). Furthermore, there is a dearth of systematic review that examines the components of CT that should be integrated into early childhood curricula. Consequently, a CT curriculum framework is highly necessary and important in the field of ECE for several reasons. Firstly, such a framework would provide clarity on the essential components to be included in the curriculum, thereby making a notable theoretical contribution and facilitating future studies conducted within a unified CT curriculum framework for ECE. Secondly, preschool teachers often lack the necessary content knowledge to effectively support children's CT learning, which hinders their progress (Strawhacker et al., 2018; Wang et al., 2020). A CT curriculum

framework for ECE would provide teachers with a comprehensive understanding of the content of CT education in early childhood settings, thus guiding teachers to integrate CT education into their classrooms. Thirdly, a CT curriculum framework for ECE holds importance for policy development. Despite the recognition of CT as a critical skill for the 21st century, many regions and countries do not currently include it in their policy documents for ECE. A refined CT curriculum framework would support the formulation of policy guidelines and promote the implementation and dissemination of CT education in early childhood settings.

In addition, no known studies have examined the PCK for programming and CT of early childhood teachers (ECTs). Teachers' pedagogical content knowledge (PCK), which "represents the blending of content and pedagogy into an understanding of how particular topics, problems or issues are organized, represented, and adapted to the diverse interests and abilities of learners, and presented for instruction" (Shulman, 1987, p. 4), has been identified as a crucial factor in predicting and improving young children's learning outcomes within specific domains (Dunekacke & Barenthien, 2021).

Finally, the complete exploration of the influence of programming tools on young children's programming learning is still lacking. Programming tools can be categorized into physical, virtual, and hybrid kits (Yu & Roque, 2019). Previous studies that investigating the effectiveness of programming on young children's CT have primarily employed physical kits, virtual kits, hybrid kits combining a physical robot with a graphical programming environment. However, to date, there is a lack of research investigating the efficacy of hybrid kits featuring virtual sprites and tangible programming environments in fostering CT skills among young children (Yu & Roque, 2019).

Based on the above analysis, the overall objective of this study is to investigate "what to teach" and "how to teach" programming and CT in the early years. Specifically, the

research aims to:

(1) Propose a CT curriculum framework for ECE that outlines the key components of CT that should be emphasized.

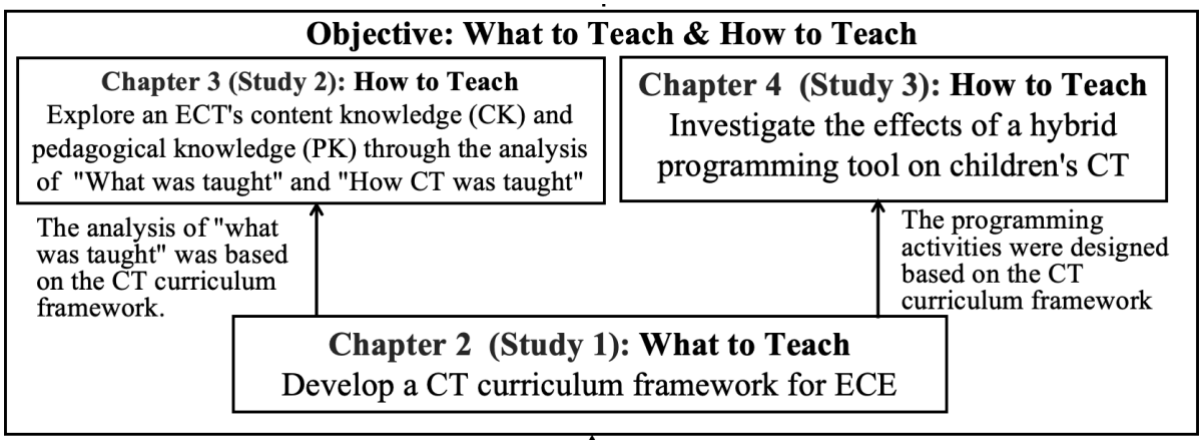
(2) Examine the pedagogical content knowledge (PCK) of early childhood teachers, particularly the content knowledge (CK) and pedagogical knowledge (PK), in the context of early programming and CT education.

(3) Examine the effectiveness of a hybrid kit with virtual sprites and tangible programming environments in promoting CT in young children.

1.8 Structure

This thesis comprises a total of five chapters. Chapter 1 offers an overview of the background that led to the conducted research, along with the objectives and structure of the thesis. Chapters 2, 3, and 4 present three separate studies, each focusing on a key issue related to CT education in ECE. Figure 1 is an overview of the three studies. Study 1 conducted a systematic review of empirical studies to establish a CT curriculum framework for ECE, thereby addressing the question of "what to teach." This framework serves as the coding framework for Study 2 and Study 3. This framework will serve as a coding framework for Study 2 and Study 3. Study 2 and Study 3 examined the question of "how to teach" programming. Study 2 investigated a preschool teacher's PCK in the field of programming and CT education, while Study 3 explored the effects of a hybrid programming tool on young children's CT. Lastly, Chapter 5 discusses the implications of the conducted research and provides suggestions for future research directions.

Figure 1 *Overview of the Three Studies*



Chapter 2: Computational Thinking in Early Childhood Education: Reviewing the Literature and Redeveloping the Three-Dimensional Framework

Yue Zeng[1 2], Weipeng Yang[2], and Alfredo Bautista[2]

[1] School of Education, Wenzhou University, Wenzhou, China; Department of Early Childhood Education, The Education University of Hong Kong, Hong Kong SAR, China

[2] Department of Early Childhood Education, The Education University of Hong Kong, Hong Kong SAR, China

Abstract

Computational thinking (CT) is gaining increasing attention from researchers and practitioners all over the world to empower children in the digital era. However, there is no consensus on which components of CT to teach beginning coders in early childhood education (ECE). To address this issue, we conducted a systematic review of 42 empirical studies focused on teaching and assessing CT in ECE. We analyzed the included studies with the three-dimensional CT framework proposed by Brennan and Resnick (2012) and demonstrated how this framework could be modified to fit the context of ECE. Based on this systematic review, we sorted out the CT components that were proven suitable for young children to learn by incorporating emerging components and removing components inappropriate for young children. We thus proposed a CT curriculum framework for ECE that covers CT concepts (i.e., control flow/structures, representation, and hardware/software), CT practices (i.e., algorithmic design, pattern recognition, abstraction, debugging, decomposition, iteration, and generalizing), and CT perspectives (i.e., expressing and creating, connecting, perseverance, and choices of conduct). This systematic review and its associated CT curriculum framework provide important theoretical contributions and practical implications for early childhood CT education.

Keywords: programming; computational thinking; early childhood teacher; content knowledge; pedagogical knowledge

Computational thinking (CT), which aligns with 21st-century skills, is vital to children's learning and development. Wing (2008) argued that CT is as important as reading, writing, and math and should be learned from the early years. Bers (2018) stated that CT assists children in becoming efficient and creative problem solvers in an increasingly digital world. In addition, aside from empowering learners to communicate with computers, CT has a significant influence on other disciplines (Wing, 2011), as well as children's metacognition (Mills et al., 2021) and self-regulation (Yang et al., 2022).

Notably, CT is gaining attention among worldwide researchers and educators in the field of early childhood education (ECE) and has been progressively incorporated into early childhood curriculum (Cho & Lee, 2017; Papadakis et al., 2016; Sung et al., 2017). Early childhood, broadly defined as ages 0-8 by the National Association for the Education of Young Children (NAEYC), is an essential period of human development. However, the answer to the question of "what to teach", which is important for all disciplines, is inconsistent across early childhood CT curricula (So et al., 2020). For example, Angeli et al. (2016) proposed a CT framework for K-6 curricula that covered five components: algorithms, abstraction, decomposition, debugging, and generalization. However, their framework was not based on empirical evidence. In contrast, the TangibleK curriculum used in some studies (Bers et al., 2014; García-Valcárcel-Muñoz-Repiso & Caballero-González, 2019) addressed sensors, sequencing, loops, branches, action-instruction correspondence, robotic motion, debugging and engineering design processes. The KIBO robotics curriculum reported in some other studies (Bers et al., 2019; Elkin et al., 2016; Pugnali et al., 2017; Relkin et al., 2021; Sullivan & Bers, 2018), however, highlighted other concepts, including sequencing, conditionals, and repeat control structures, while the card-coded robotics curriculum used in Nam et al. (2019) study involved the CT skills of sequences, representation, and being iterative and incremental.

Moreover, there is no systematic review of what components of CT should be embedded in early childhood curricula based on empirical studies. To address this knowledge gap, this review aims to propose a CT curriculum framework for ECE articulating what CT components a curriculum should promote, by systematically examining relevant empirical studies targeting children aged 2-8.

2.1 Defining CT

To establish a CT curriculum framework for ECE, working through the definition of CT is necessary. Wing (2006) coined the term CT and viewed it as a way of "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). A few years later, Wing (2010) updated the definition of CT as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (p. 1). CSTA and ISTE (2011) developed an operational definition of CT, which refers to a problem-solving process, covering core skills such as abstraction, problem reformulation, logical thinking, algorithmic thinking, selecting the optimal solution, generalization and problem transfer. These skills are further supported and enhanced by learning dispositions, such as confidence, persistence, collaboration, tolerance for complexity, and the ability to deal with open-ended problems. Shute et al. (2017) defined CT as "the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts" (p. 151) and categorized CT into six aspects: decomposition, abstraction, algorithm design, debugging, iteration, and generalization. In contrast, Bers et al. (2019) viewed CT not only as a problem-solving ability but also as "an expressive process that allows for new ways to communicate ideas" (p. 131). Bers (2018) described seven key components of CT for children aged 4 to 9, including control structures, representation,

hardware/software, algorithms, modularity, debugging, and design process.

As described above, there is not one unanimous definition or model of CT. While classical, Wing's (2006, 2011) definitions are relatively broad. Definitions by other institutions or researchers are relatively operational, but these definitions or frameworks vary in the dimensions of CT. For example, the definition of CSTA and ISTE (2011) involves CT practices (problem-solving process) and CT perspectives (learning dispositions). Differently, Shute et al.'s (2017) definition entails a set of CT practices, while Bers's (2018) definition includes CT practices (algorithms, modularity, debugging and design process) and CT concepts (control structures, representation, hardware/software). Instead, the present systematic review will propose a CT curriculum framework for ECE to embrace CT concepts, practices, and perspectives as an organic system. Building upon the commonalities of the above definitions (i.e., problem solving), we define CT as an approach to solving problems that are often messy, complex and open-ended in various disciplines, with the use of computational concepts, practices, and perspectives.

2.1.1 Previous reviews on CT in school education

To the best of our knowledge, 20 reviews have examined CT in the school context. Among them, 17 reviews have investigated CT in K-16 schools, and three have examined CT in ECE.

Researchers have examined CT in K-16 education from five main facets: CT mapping, CT definition and model, CT teaching and learning, CT assessment, and interrelations between CT and creativity. Tikva and Tambouris (2021) developed a conceptual model which described different CT research areas and their relationships. Shute et al. (2017) established a model of CT based on a review of theoretical work. Ezeamuzie and Leung (2022) proposed a CT model emphasizing algorithmic solutions drawing on programming concepts. Zhang and Nouri (2019) investigated the empirically supported CT

abilities that can be learned using Scratch in K-9. Regarding CT teaching and learning, two reviews examined teaching CT through programming (Lye & Koh, 2014; Sun et al., 2021a). Sun et al. (2021b) and Zhang et al. (2021) separately investigated the effectiveness of educational games and robots for improving students' CT. Huang and Looi (2021) and Lee et al. (2022) separately examined how "unplugged" pedagogies and CS education enhance students' CT skills. Other researchers focused on integrating CT into the school curriculum (Chan et al., 2022; Kite et al., 2021; Ogegbo & Ramnarain, 2021; Wang et al., 2021). In addition, two reviews specifically investigated the assessment of CT (Cutumisu et al., 2019; Tang et al., 2020). Furthermore, Israel-Fishelson and Hershkovitz (2022) explored the inter-dependencies between creativity and CT.

In the context of ECE, Bakala et al. (2021) examined robot-mediated activities to foster preschool children's CT. McCormick and Hall (2021) examined CT learning experiences design, educational outcomes, and CT research design. Bati (2021) investigated whether the variables of plugged-in versus unplugged, gender, and age affect CT teaching and learning in early childhood.

Among the aforementioned 20 reviews, three of them (Ezeamuzie & Leung, 2022; Shute et al., 2017; Zhang & Nouri, 2019) have proposed a CT framework or model. However, they were targeted at K-16 education without focusing on a significantly different stage of education-ECE. Based on a meta-review of the existing reviews, we found that no review has focused on what CT components can be embedded in ECE. A robust CT curriculum framework for ECE will provide researchers and teachers with a better knowledge of the content of CT education in early childhood settings. Nevertheless, there is no such framework for CT in ECE.

2.1.2 The three-dimensional CT framework

In 2012, Brennan and Resnick developed a groundbreaking CT framework for studying and assessing CT. Their framework includes three essential dimensions: CT concepts, CT practices, and CT perspectives. CT concepts are the concepts children need to master to understand the mechanics of programming, including sequences, loops, parallelism, events, conditionals, operators, and data (Brennan & Resnick, 2012). CT practices are skills and strategies applied by children while solving problems, which include four main sets: being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing (Brennan & Resnick, 2012). CT perspectives are the learning dispositions displayed by children when programming, including expressing, connecting, and questioning (Brennan & Resnick, 2012). This systematic review will adopt Brennan and Resnick's (2012) three-dimensional framework to identify concrete CT components that can be embedded in ECE. The rationales for using this framework are explained below.

First, the three-dimensional framework provides a comprehensive and integrated theoretical framework to frame the components of CT education. Kong (2016) stated that Brennan and Resnick's (2012) CT framework offered "a wide coverage of CT" (p. 379). Many curricula focused on teaching what Brennan and Resnick (2012) referred to as "CT concepts" (Falloon, 2016); however, rather than only emphasizing CT concepts, Brennan and Resnick's (2012) framework also highlights the process of thinking and learning, i.e., CT practices, and the social attribute of CT, i.e., CT perspectives (Allsop, 2019; Zhong et al., 2016). The three dimensions, just like a cube's length, width and height, are inextricably and organically combined as children's CT learning content and outcomes.

Second, Brennan and Resnick's (2012) CT framework has been used as a basis to identify CT components and evaluate the CT learning outcomes in previous studies. Lye and Koh (2014) used Brennan and Resnick's (2012) CT framework to review intervention studies

in K-12 contexts. They found that CT concepts were the focus of most studies and suggested that future studies should concentrate more on CT practices and perspectives. Zhang and Nouri (2019) systematic review showed that Brennan and Resnick's (2012) framework could capture the CT skills K-9 students gained from using Scratch to a great extent. Therefore, teachers and researchers can use the framework when planning lessons or designing projects. Chalmers (2018) adopted Brennan and Resnick's (2012) CT framework as a data analysis framework to investigate Australian primary school teachers' perceptions of what students learned in robotics-based STEM activities and identified three core themes of computational concepts, practices, and perspectives. They further indicated that it would be more effective for teachers to incorporate CT into the primary curriculum if they had a deeper understanding of CT concepts, practices, and perspectives. Nouri et al. (2019) interviewed Swedish teachers to understand what CT skills they perceived K-12 students developed when learning programming. They identified three themes related to CT skills that corresponded well with the three dimensions described by Brennan and Resnick (2012). In addition to investigating what components of CT students learned, researchers also adopted Brennan and Resnick's (2012) framework to assess students' CT development. For instance, Zhong et al. (2016) adopted the framework to develop the Three-Dimensional Integrated Assessment framework to assess computational concepts, practices, and perspectives comprehensively.

Third, consistent with our definition of CT, the three-dimensional CT framework, initially proposed for conceptualizing CT in the context of programming with Scratch, can be adapted for other learning contexts (Brennan & Resnick, 2012). For example, the CT practice of decomposition can be learned or applied in children's daily routines, such as getting ready for school, washing hands, toileting, and making bread.

However, as Brennan and Resnick's (2012) framework was constructed based on Scratch-based activities of young people aged 8–16 years, some components of this CT

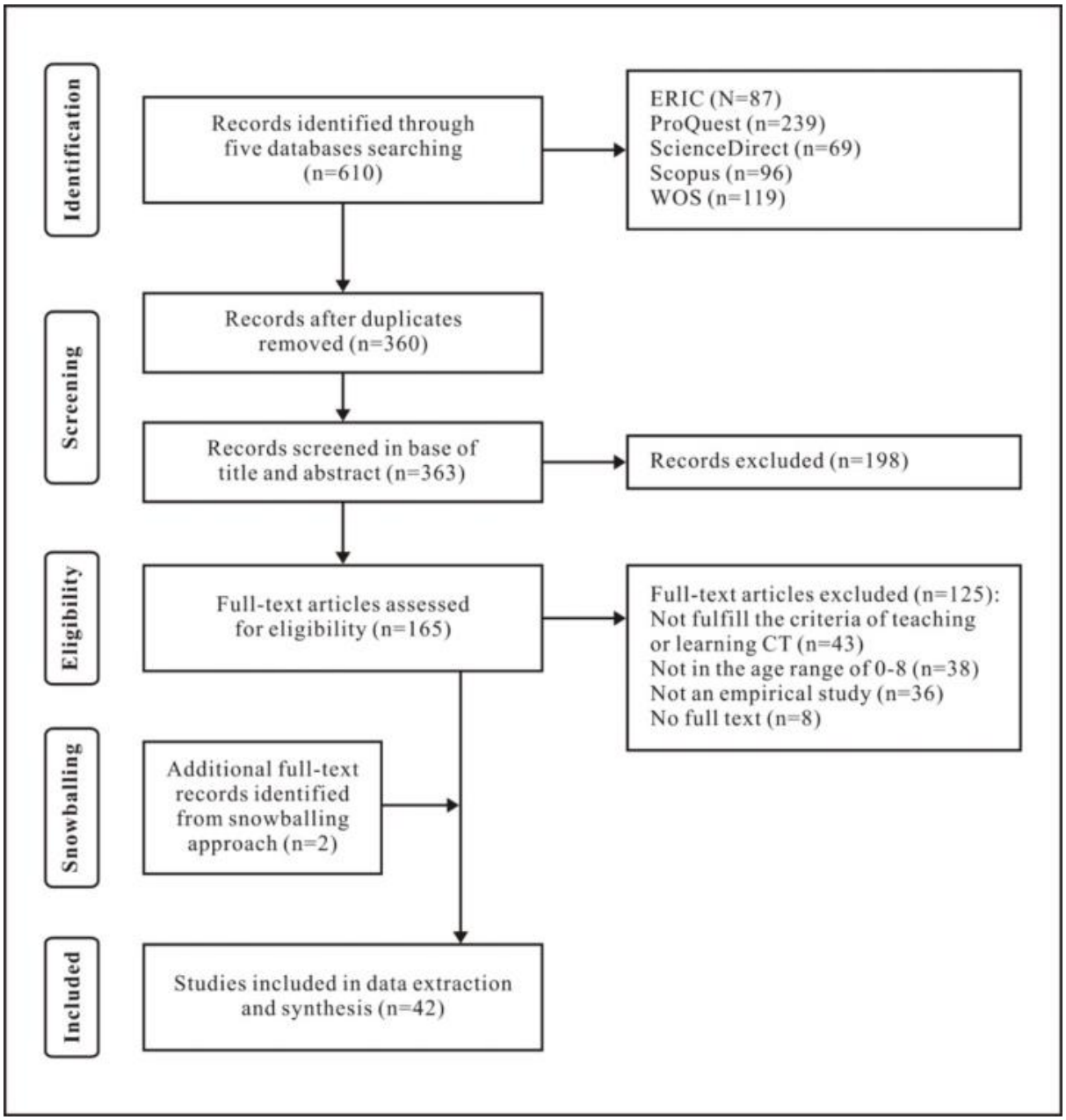
framework may not be age-appropriate for children below 8. It needs to be refined to fit the context of ECE.

Informed by Brennan and Resnick's (2012) three-dimensional CT framework, we conducted this systematic review with the primary goal of establishing a CT curriculum framework for ECE. Specifically, this systematic review of CT studies in the field of ECE aims to address this overarching question: *Which CT components involved in the empirical studies have been included in the three-dimensional CT framework, and which are newly emerging?* Based on the findings, we further propose an early childhood CT curriculum framework.

2.2 Method

To comprehensively gather, evaluate, and synthesize existing evidence and ensure review procedures' reliability, validity, and reproducibility, we conducted a systematic review (See Figure 2). A systematic review protocol was developed by the first author and reviewed by the second author, who acted as the auditor in advance to pre-specify the objectives and approaches of the systematic review (Liberati et al., 2009).

Figure 2 *PRISMA Diagram for the Search and Selection Processes*



2.2.1 Literature search

We thoroughly searched five widely used digital databases, namely, Web of Science, SCOPUS, ProQuest, ERIC, and ScienceDirect, to ensure that the search covers all literature related to CT in early childhood.

First of all, the first author defined the keywords and carried out the search. Two keywords, "computational thinking" and "early childhood," relevant to the review and their synonyms were identified. Although CT is related to computing, programming and coding,

like Tang et al. (2020), we did not use these terms as alternative keywords because CT refers to an approach to solving problems computationally rather than the ability to program (i.e., programming skills) (Macrides et al., 2021; Voogt et al., 2015). To specifically focus on CT, we reviewed studies that explicitly used the term "computational thinking". Meanwhile, we used preschool*, kindergarten*, pre-K*, prekindergarten*, "early child*" "early age*", "early years", "young child*", "young learners", child*, "elementary education", "lower education", "primary education", "pre-primary education" as synonyms for "early childhood".

In terms of the search filters, we set the start of the timeline as the year 2006, when Wing (2006) first used the term CT in her seminal article, signaling the beginning of a new area of research. Notably, the literature search was conducted at the end of October 2021. In addition, the search scope was limited to papers published in English and available in full text. Regarding the literature type, we included peer-reviewed articles, conference papers, books, and dissertations. Detailed information about the search strings, search parameters, search date, and the number of items found is given in the Appendix.

2.2.2 Inclusion and exclusion criteria

After completing the literature search, we deleted the duplicate papers caused by the same papers appearing in different databases. We then applied the following inclusion and exclusion criteria to identify the eligible papers.

Game-Based Unplugged and Plugged-in Activities in Primary School" (only programming curriculum were introduced) were excluded.

Inclusion Criteria (ICs) include the following:

IC1. The research should focus on teaching, learning, or assessing CT.

IC2. We only focused on children's CT development in the school scenario, regardless of the context being formal or informal.

IC3. Participants of the research were children aged 2–8. Those papers whose participants were partly in this age group or part of the activities (e.g., a separate subsection) about children in this age group can also be included in this review.

IC4. The paper reports an empirical study. Included studies must report the methodology, participants, and findings specifically.

Exclusion Criteria (ECs) include the following:

EC1. CT was not specifically focused on, such as papers examining the effect of CT learning on children's cognitive skills or other abilities, or papers only addressing robotics, artificial intelligence, kindergarten information/computer technology, or mathematics learning and teaching.

EC2. The study focused on parental influence on children's CT or parent-child interaction in CT learning, such as "Parental influence on children's computational thinking in an informal setting," "Examining the role of parents in promoting computational thinking in children: a case study on one homeschool family," and "Parent-child interaction and children's learning from a coding application".

EC3. Only programming tools or curricula are introduced without empirical data in the article. For example, "Robots and Robotics Kits for Early Childhood and First School Age" (only programming tools were introduced) and "Training Computational Thinking: Game-Based Unplugged and Plugged-in Activities in Primary School" (only programming curriculum were introduced) were excluded.

There are several rationales for the inclusion and exclusion criteria. First, because our goal was to develop a CT curriculum framework for teaching and learning in ECE, we only included studies related to the teaching and learning of CT. Second, studies on CT assessment need to be included since they involve measuring children's learning outcomes which are the components of CT. Third, since our goal was to create a CT curriculum framework to inform

CT education in ECE, we only focused on studies conducted in school contexts, and the studies on parental influence on children's CT or parent-child interaction in CT learning were excluded. Fourth, because we targeted the early childhood stage and found in the literature search that the youngest age of children in related studies was 2 years old, we determined the age range as 2–8 years old. Fifth, our goal was to develop a CT curriculum framework, and the approach to establishing such a framework is to incorporate evidence-based CT components. Therefore, the eligible papers must be empirical studies.

According to the selection criteria, the first two authors independently screened the titles and abstracts of the records. We then compared the included papers and calculated Cohen's kappa coefficient to measure the inter-rater reliability (Cohen, 1960). The coefficient was 0.93, which indicated a highly satisfactory level of agreement (McHugh, 2012). We downloaded and read the full text of papers classified differently between reviewers to decide on inclusion/exclusion. Specifically, the first two authors analyzed the questionable items, and differences were resolved through discussion. After completing the search and selection process, we identified 40 studies.

2.2.3 Snowballing

To further reduce the probability of missing relevant studies, we conducted the snowballing selection procedure. Four well referenced literature reviews were used as the snowballing seeds. These four literature reviews are respectively about CT assessment, robot-mediated activities to foster CT, CT learning experience design and evaluation, and the relationship between CT and coding experiences (see the Appendix for more details). The search of papers in Google Scholar yielded 150 citations based on forward snowballing and 502 references based on backward snowballing (652 papers in total). Using the eligibility criteria above, we found two new papers.

2.2.4 Data extraction and synthesis

The data extraction and synthesis process included 42 papers in total (See Figure. 1). The three-dimensional CT framework (Brennan & Resnick, 2012) was used as the coding framework (See the Appendix). The first two authors reviewed and coded 11 identical randomly chosen articles (about 25% of all articles). The interrater reliability reached 0.90, and disagreements were solved through discussion. The first author then coded the remaining papers independently as the interrater reliability was high (McHugh, 2012).

2.3 Results

2.3.1 Overview of the included studies

An overview of the included studies can be found in the Appendix, covering information about the authors, country, participants' age, sample size, research methods, CT assessment instrument(s), intervention tool(s), and intervention duration.

The included studies were all published from 2013 to October 2021, while the most frequent publication years were 2019 (9) (The number in parentheses represents the number of papers), 2021 (8) and 2020 (7). Among the 42 papers, around half of the studies (22) were conducted in the USA, and the rest were scattered in Spain (4), the UK (2), Australia (2), Hong Kong (2), Mainland China (1), Singapore (1), Netherlands (1), Greece (1), Turkey (1), the Republic of Korea (1), Uruguay (1), and Cyprus (1). Two papers do not specify a concrete country, only stating their location as the Midwest and Southern European Country. Notably, more than 20% of the studies were conducted by Bers and her team at Tufts University's DevTech Research Group (9).

The articles include a variety of preschool-age subgroups, with the age range of 2–8 years old. All but four of the papers had multiple age groupings. There were 15 papers with a sample size of fewer than 30 children and 27 papers with a sample size of 30 or more. A study by Relkin et al. (2021) had the largest sample size (848 children). Three studies had a sample of only three children (García-Valcárcel-Muñoz-Repiso & Caballero-González, 2019;

Metin, 2020; Wang et al., 2020). Of the 42 papers, 25 used quantitative methods, 10 used mixed methods, and 7 used qualitative methods.

Different CT learning tools were found in the literature. These CT learning tools can be classified into four categories: tangible robotics, digital or screen-based applications, unplugged kits and hybrid kits. More than half of the studies (22) in this systematic review used tangible robotics to develop children's CT. There were 11 different kinds of robots (two of the studies did not specify the type of robots) out of the 22 studies, with the most significant number of studies using Bee-Bot robotics (6), followed by KIBO (5). Eight studies used digital or screen-based applications, and half used ScratchJr (4). Seven studies used hybrid kits, and all these hybrid kits used CHERP as the programming software. Four studies designed unplugged CT activities, and one used both robotics and digital or screen-based applications. Studies also used different CT measurements, which can be broadly divided into five categories (see Table 1).

Table 1 *Types of CT Measurements*

Types	Description of the method	The use of these methods in studies
Test (5 studies)	A test consisting of single or multiple choice, fill-in-the-blank or open-ended questions, usually evaluated by completeness and correctness (Tang et al., 2020)	TechCheck (Relkin et al., 2021) Paper-based programming skill test (Ahn et al., 2021; Sung & Black, 2021) Test consisting of items from the "International Bebras Contest" (del Olmo-Muñoz et al., 2020) Test adapted from Tran's (2019) CT questionnaire (Gerosa, 2021)
Project or task assessment	Evaluation of children's projects or performances	Solve-Its task-based assessment (Bers et al., 2019; Elkin et al., 2016; Pugnali et

(21 studies)	during CT tasks	al., 2017; Strawhacker & Bers, 2015; Strawhacker & Bers, 2019; Strawhacker et al., 2018; Sullivan & Bers, 2016; Sullivan & Bers, 2018) The robot and/ or program assessment with a scale (Bers et al., 2014; Pila et al., 2019; Saxena et al., 2020; Sullivan & Bers, 2013; Sung et al., 2017) Story/ picture sequencing task (Kazakoff et al., 2013; Nam et al., 2019) The "SSS" rubric used in the TangibleK program (Muñoz-Repiso & Caballero-González, 2019) The "Hokey-Pokey" program completeness assessment rubric (Flannery & Bers, 2014) The Coding Development (CODE) Test 3-6 (Critten et al., 2021) Korean version (Ryu, 2003) of Ward's (1993) original problem-solving performance instrument (Nam et al., 2019) Self-developed CT rubric (Angeli & Valanides, 2020; Georgiou & Angeli, 2019)

		Evaluation based on the number of CT tasks completed and the time taken to complete them (Rijke et al., 2018)
Classroom observation (8 studies)	Observe children's behavior during CT activities with a checklist	The PTD checklist (Bers et al., 2019; Pugnali et al., 2017; Sullivan & Bers, 2018) Children communication checklist (Critten et al., 2021) Self-developed CT behavior observation system (Terroba et al., 2021) Checklist of behaviors drawing upon Bird and Edwards (2014) (Newhouse et al., 2017) The Basic Coding Skills Observation Form and the Basic Robotic Coding Skills Observation Form (Metin, 2020) CT rubric designed by Leonard et al. (2016) (Qu & Fok, 2021)
Interview (1 study)	Interview the children while they are performing a problem-solving task	Moore et al. (2020)
Self-evaluation (1 study)	Self-evaluation of skill level after class	Cho and Lee (2017)

According to the results of this systematic review, previous studies have examined different components of CT unevenly, as shown in Table 2 (See the Appendix for more details about CT concepts, practices and perspectives emphasized by each study). In the 42 reviewed literature, the number of papers studying CT concepts is the largest, followed by CT practices and CT perspectives. Below, we report the CT components involved in the studies.

Table 2 *Frequency of Each CT Component in the Included Studies*

CT Concepts	**CT Practices**	**CT Perspectives**
Classic CT Concepts	***Classic CT Practices***	***Classic CT Perspectives***
Sequences (31 studies)	Testing and debugging (23 studies)	Connecting (15 studies)
Loops (18 studies)	Modularizing/ Decomposition/	Expressing (12 studies)
Events (16 studies)	Problem reformulation (16	Questioning (0)
Conditionals (10 studies)	studies)	
Parallelism (1 study)	Abstraction (7 studies)	
Operators (0)	Being iterative and incremental/	
Data (0)	Design process (6 studies)	
	Reusing and remixing (0)	
Emerging CT Concepts	***Emerging CT Practices***	***Emerging CT***
Representation (9 studies)	Algorithmic Design (13 studies)	***Perspectives***
Control flow/ structures (4	Pattern recognition (7 studies)	Choices of conduct (4
studies)	Generalizing (2 studies)	studies)
Hardware/ Software (4	Logical thinking (2 studies)	Perseverance (2 studies)
studies)	Simulations (1 study)	
Automation (1 study)	Spatial reasoning (1 study)	

2.3.2 Classic CT concepts

Although Brennan and Resnick's (2012) framework includes seven concepts, ECE researchers mainly focused on the concepts of sequences (31 papers, 73.8%), loops (18 papers, 42.86%), events (16 papers, 38.1%) and conditionals (10 papers, 23.81%). The concept of parallelism is only briefly mentioned in one study when introducing the programming intervention (i.e., Gordon et al., 2015). No studies mentioned the concepts of data and operators. The mainly focused CT concepts are explained below one by one.

2.3.2.1 Sequences

Sequences is that "a particular activity or task is expressed as a series of individual steps or instructions that can be executed by a man or a computer" (Brennan & Resnick, 2012, p. 3). Sequences learning activities in the reviewed studies can be broadly categorized into robotic activities, graphical programming activities, and unplugged activities. In robotic activities, children are usually asked to program a robot to complete pre-designed tasks. For example, in Angeli and Valanides's (2020) study, children were asked to program the Bee-Bot to leave the hive, collect and carry pollen from flowers of a specific color, and visit the Bee-Bot's friends before returning to the hive. In Bers et al.'s (2014) study, one task was to program the KIBO to dance. In graphical programming activities, children are usually asked to complete tasks or create their projects in a graphical programming environment (e.g., Code.org, ScratchJr) by creating programs to control virtual characters on the screen to move (Del Olmo-Muñoz et al., 2020). In unplugged activities, different approaches are used to teach or learn sequences. One way is like the robotics activity, but instead of programming a robot, the students themselves act like robots and "programmed" by their partners or teachers (Critten et al., 2022; Saxena et al., 2020). Students could also manipulate an object to follow the arrows or walk on a map with their fingers (Critten et al., 2022). Another way is to get

children to do something in the correct order, e.g., dressing the baby (Critten et al., 2022) or sequencing pictures correctly (Saxena et al., 2020).

All these 31 studies confirmed that young children could master the concept of sequences. Elkin et al. (2016) found that three-year-old children could program KIBO robots in the syntactically correct order, and Critten et al. (2022) stated that even children aged two years old could learn sequences through play-based learning practices. However, research indicated that older preschoolers (about five years old) significantly outperformed younger preschoolers (under five years) on "Easy sequencing" and "Hard sequencing" (Elkin et al., 2016; Sullivan & Bers, 2013, 2016). In addition, it was more challenging to sequence a more extended program than a shorter one, even though both tasks utilized the same programming concepts (Elkin et al., 2016; Sullivan & Bers, 2016).

2.3.2.2 Events

Events is "one thing causing another thing to happen" (Brennan & Resnick, 2012, p. 4). Events was often described as "action-instruction correspondence" (Bers et al., 2014; Flannery & Bers, 2013; García-Valcárcel-Muñoz-Repiso & Caballero-González, 2019; Sullivan & Bers, 2013). Researchers also referred to events as "trigger-action relationships"; for example, a child used a flashlight to trigger a light sensor (Sullivan & Bers, 2016) or a child interacted with the robot by triggering the rules (Gordon et al., 2015).

To teach children about events, teachers taught cause and effect when they pressed buttons on a programming platform (Sullivan & Bers, 2013) or encouraged children to press different buttons and observe the results (Angeli & Valanides, 2020; Bers et al., 2019). The most preliminary learning of events occurs when children learn the relationship between cause and effect. Children, however, may randomly input commands when they are unable to observe and interpret this relationship (Newhouse et al., 2017). Therefore, events is a crucial concept that children need to master in CT learning (McCormick & Hall, 2021).

2.3.2.3 Loops

Loops refers to repeating the same instruction multiple times (Brennan & Resnick, 2012). In some studies, researchers equate loops with repeat (Elkin et al., 2016; Pila et al., 2019; Sullivan & Bers, 2018), while in others, researchers argue that loops is different from repeat in that students first identify repeated patterns and then use loops to represent this repeat (Del Olmo-Muñoz et al., 2020).

Like sequences, children usually learn loops in robotic, graphical programming, or unplugged activities. For example, one robotic activity asked children to practice estimation to choose the correct numerical parameters needed to make their robot travel a certain distance (Elkin et al., 2016). One of the graphical programming activities supported students in understanding loops when asking students to use loops to help the bee collect more nectar (Del Olmo-Muñoz et al., 2020). Moreover, one of the unplugged activities required children to find repeated patterns in the code and write algorithms to represent these patterns with loops (Del Olmo-Muñoz et al., 2020).

Compared to sequences and events, loops is a more advanced concept. Elkin et al. (2016) found it challenging for children aged 3–5 to understand loops. Because loops involve not only keeping a new piece of syntax in their working memory but also requires children to make mathematical estimates and reason with numerical parameters (Elkin et al., 2016). (Sullivan & Bers, 2016) found that the first and second graders could spend time using loops to create complex programs for their robots while the pre-kindergarteners could not. In Bers et al.'s (2019) KIBO curriculum, understanding loops is the learning objective only for children older than four. Pila et al.'s (2019) study also involved learning about loops, but the children who participated were all over four years old.

2.3.2.4 Conditionals

Conditionals is "the ability to make decisions based on certain conditions, which supports the expression of multiple outcomes" (Brennan & Resnick, 2012, p. 5). Like loops, conditionals is also a challenging concept for children. Pila et al. (2019) used two tablet-based apps to teach conditionals, but the children in this study were all at least four years old, and their CT knowledge on sequencing and loops all significantly increased except for conditionals. Strawhacker and Bers (2015) also found that 5–6 years old children had trouble learning conditionals. In Bers et al.'s (2019) study, the learning goal of understanding conditional instruction was only for children aged five, and in Relkin et al.'s (2021) study, conditional statements were included in the second-grade curriculum, not in the first grade. However, contradicting these results, other researchers demonstrated that 3–6 years old children could have an excellent understanding of Conditional If Statements (Sullivan & Bers, 2018) and four-year-old children can master conditional statements (Kazakoff et al., 2013; Sullivan & Bers, 2016).

2.3.3 Emerging CT concepts

Some concepts appeared in the literature but were not covered in the three-dimensional CT framework, including representation (9 papers, 21.43%), control flow/control structures (4 papers, 9.52%), hardware/software (4 papers, 9.52%) and automation (1 paper, 2.38%).

2.3.3.1 Representation

The notion that symbols represent concepts is essential for early learning, including reading and mathematics (Bers, 2018). Although many studies do not directly mention "representation", the learning of the concept of representation is prevalent in CT experiences. Because when children "program" an object's behavior, they must understand that each

instruction represents an action to be taken by an object (Bers, 2018; Bers et al., 2014; Relkin et al., 2021).

Murcia and Tang (2019) indicated that iconic representations are crucial in making abstract concepts or symbolic representations easier for children to understand. Nam et al. (2019) provided worksheets to have children create a visual representation of their solution before coding. Likewise, Critten et al. (2022) provided photographs, symbols, and images as codes to form algorithms in route planning and coding activities. Moore et al. (2020) examined how children resolved CT tasks by translating between representations and found that children used concrete representations to simplify the translation, language as a scaffold between translations, and concrete actions to represent or assist the translation. Relkin et al.'s (2021) study suggested that representation improved significantly after the CAL-KIBO curriculum.

2.3.3.2 Control flow/structures

The terms "control flow" and "control structures" share the same meaning. Control flow/structures refers to the concept that a programmer can control the order in which a robot follows to achieve a goal (Bers et al., 2014). Control flow/structures determine the order in which instructions are followed or executed in a program (Bers, 2018). Control flow/structures involve some other CT concepts, such as sequences, loops, conditionals, and events (Bers, 2018).

Researchers found that older students can use more complex control structures than younger children, probably because older students are more likely to understand complex programming blocks (Portelance et al., 2016). Similarly, Strawhacker and Bers (2019) found that while 5–6 years old children struggled with the concept of control flow, first and second graders would spend more time working on these complex instructions and strategies to program stories and games.

2.3.3.3 Hardware/software

Software and hardware work in tandem to perform tasks (Bers, 2018). The software provides instructions to the hardware and the hardware receives and executes these instructions (Bers, 2018). Students can understand this relationship between hardware and software during the programming process. There are also other ways to learn the concept of hardware/software, such as playing games about what a robot is and is not (Relkin et al., 2021), exploring the basic robotic parts of the robot (Sullivan & Bers, 2016), and learning the Robot Parts song (Elkin et al., 2016).

2.3.3.4 Automation

Automation is to achieve the automatic operation of a process or system (Shute et al., 2017). By inputting algorithms or programs into machines/computers and seeing machines/computers execute the algorithms or programs, children can understand the concept of automation. However, in the reviewed studies, only Khoo (2020) explicitly examined automation and illustrated automation with the OZO-Bot.

2.3.4. Classic CT practices

Testing and debugging (23 papers, 54.76%) is the CT practice that arises most often in the literature, followed by decomposition (16 papers, 38.10%), abstracting (7 papers, 16.67%), and being iterative and incremental (6 papers, 14.29%), while reusing and remixing is not examined.

2.3.4.1 Testing and debugging

Testing and debugging is the skill of identifying and addressing problems that impede task completion (Bers et al., 2014; Georgiou & Angeli, 2019; Moore et al., 2020). In some studies, it was referred to as “problem-solving” (Gerosa et al., 2021) and “trouble shooting” (Bers et al., 2014; Ehsan et al., 2021; Sullivan & Bers, 2013, 2016). The debugging process consists of four steps: (1) recognize that something is wrong, (2) maintain the initial goal or

change to a suitable alternative, (3) make assumptions about the reason for the problem, and (4) try to solve the problem (Bers et al., 2014; Sullivan & Bers, 2013).

According to studies, explicit instructions and scaffolding are necessary for children to master debugging skills (Newhouse et al., 2017; Terroba et al., 2021). One of the key strategies is to provide pre-fixed errors that children need to identify and fix before they can do the natural debugging that occurs during the programming activities (Bers et al., 2014; Gerosa et al., 2021; Wong & Jiang, 2018).

Other strategies included: (1) reminding kids to stop and evaluate, (2) modeling the process of detecting errors, (3) motivating children to explore different approaches (Wang et al., 2020), and (4) discussing potential solutions with peers (Sullivan & Bers, 2013). Critten et al. (2022) found that a friendly and informal approach effectively fostered team cohesion and encouraged children to find and fix errors.

Studies show that plugged programming tools could support debugging learning because they allow students to observe the actions of virtual characters or robots; when the objects do not move as expected, children would find out the error and try to modify the program (Gerosa et al., 2021; Moore et al., 2020; Qu & Fok, 2021). However, Pugnali et al. (2017) found that children aged 4–7 in the KIBO robot group showed significantly better debugging skills than the ScratchJr group; they explained this is because, in the early stages of development, children depend on interacting with physical objects to learn. Children in the KIBO group could physically manipulate blocks and observe the robot's motion in physical space. Children can also practice their debugging skills in unplugged activities. Ehsan et al. (2021) observed children's behavior in the engineering design process and found that children exhibited debugging in the design evaluation and revision process.

Studies confirmed that children's debugging skills improved after learning CT curricula. Bers et al. (2014) discovered that children could partially to mostly understand and

apply debugging skills after the TangibleK curriculum. Bers et al. (2019) found that children (3–5 years old) scored highly on debugging skills after the KIBO curriculum. Strawhacker et al. (2018) found that children aged 5 to 8 could all grasp the debugging skill after the ScratchJr curriculum.

2.3.4.2 Decomposition/problem reformulation

Decomposition, or problem reformulation, refers to "the skill of reformulating a difficult problem as a familiar one or breaking it down into smaller parts in order to make the problem easier to solve" (Wang et al., 2020, p. 4). Plugged and unplugged activities were employed to foster decomposition skill in the reviewed studies. One example of the plugged decomposition task is programming simulations of storybook characters (Relkin et al., 2021). One example of the unplugged decomposition task is making as many decompositions of dance movements as possible to make it clear that other students can dance with only the design sheet (Rijke et al., 2018). Wang et al. (2020) summarized strategies to scaffold children's decomposition skills, such as linking problems to things children are familiar with, modeling decomposition, providing verbal hints by thinking aloud and embodied instruction with gestures and body movements.

Studies also found that young children were able to cope with complex problems through decomposition (Angeli & Valanides, 2020; Dietz et al., 2019; Murcia & Tang, 2019). For example, Ehsan et al. (2021) found that children (7–8 years) decomposed the overall problem (building a space for a puppy) into smaller tasks, which helped them identify the main parts of the task and the key criteria.

2.3.4.3 Abstraction

Abstraction is considered the essence of CT (Wing, 2008); it allows people to simplify and manage complexity by focusing on relevant information (and discarding irrelevant detail) to identify patterns and commonalities among different representations.

Researchers argued that early exposure to abstraction during kindergarten is necessary (Gibson, 2012; Khoo, 2020).

Both plugged and unplugged activities are beneficial to developing children's abstraction ability. Regarding plugged activities, Qu and Fok (2021) found that student-robot interactions play a critical role in children's abstraction since the robot serves as a visible and tangible agent between the real and abstract worlds. In other words, children's interaction with the robot allows them to transfer the real-world situation to the process of programming. Regarding unplugged activities, in Moore et al.'s (2020) study, children were asked to translate concrete objects on the physical route into abstract representations on the map. In another unplugged abstraction task, the students paired up and received cards containing words to portray. When portraying the objects, children had to abstract the most important details of the concept and ignore unimportant details (Rijke et al., 2018).

2.3.4.4 Being iterative and incremental/(engineering) design process

Seven studies have integrated "being incremental and iterative" by teaching students to build artifacts using the Engineering Design Process (EDP). The EDP is an iterative design process that engineers use to design products to satisfy specific requirements (Bers et al., 2014; Sullivan & Bers, 2013). Rather than expecting immediate success or getting it right the first time, it stresses the need to keep working on and improving the work without giving up and accept failure as part of learning (Bers et al., 2014; Sullivan & Bers, 2013). The EDP is adapted for ECE by Bers (2018) into six steps: asking, imagining, planning, creating, testing and improving, and sharing. In the TangibleK curriculum, the EDP was a central component; it was introduced in the first class and practiced throughout the course (Bers et al., 2014; Sullivan & Bers, 2013). In the KIBO and CAL-KIBO robotics curriculum, children were required to create a final KIBO project applying the EDP (Bers, 2019; Elkin et al., 2016; Relkin et al., 2021).

2.3.5 Emerging CT practices

There are six CT practices not included in the three-dimensional CT framework, namely algorithms (13 papers, 30.95%), pattern recognition (7 papers, 16.67%), generalizing (2 papers, 4.76%), logical thinking (2 papers, 4.76%), simulation (1 paper, 2.38%), and spatial reasoning (1 paper, 2.38%).

2.3.5.1 Algorithmic design

The defining features of algorithmic design are as follows: (1) Algorithmic design is to solve a specific problem or complete a task (Khoo, 2020; Shute et al., 2017); (2) Algorithmic design is composed of a series of ordered steps (Khoo, 2020) that involve not only the most basic sequential concepts (Angeli et al., 2016; Shute et al., 2017) but also other concepts such as loops, conditionals, and parallelism (Lu & Fletcher, 2009; Qu & Fok, 2021); (3) A computer or human can carry out the instructions (Shute et al., 2017); (4) Algorithmic design is related to the efficiency of creating optimal solutions and automation (Shute et al., 2017).

Scaffolding is necessary for young children to learn algorithmic design. Newhouse et al. (2017) found that students were unlikely to exhibit any actions indicative of understanding algorithmic design without explicit scaffolding. As an example of algorithmic design activities, Shute et al. (2017) shared a maze activity in which students had to find the shortest route that met specific criteria. Angeli and Valanides (2020) proposed two methods to develop children's algorithmic design skills, and they found that boys benefited more from individualistic, kinesthetic, space-oriented, and action-based card activities, while girls benefited more from collaborative writing activities. In Khoo's (2020) study, children worked collaboratively, figured out the algorithmic design on the worksheets and then tested the Mouse Robot and compared it with their answers. In addition to robot programming activities, researchers also used unplugged activities to teach algorithmic design. In Critten et

al.'s (2022) study, children were asked to place pictures of clothes in order of dressing, and if there were mistakes, they were encouraged to figure out together the correct order of the algorithms. To extend children's algorithmic design skills, teachers can increase the difficulty of the directional game (e.g., adding additional obstacles/treasures) (Saxena et al., 2020) and the complexity of the algorithms (Angeli & Valanides, 2020).

2.3.5.2 Pattern recognition

Pattern recognition is "observing patterns, trends, and regularities in data" (Hsu et al., 2018, p. 25). The skill of pattern recognition is related to the concept of loops. To use loops to repeat patterns, one must identify repeated patterns first (Del Olmo-Muñoz et al., 2020). Pattern recognition is also the sub-processes of abstraction (Shute et al., 2017).

Researchers found that hands-on practice is a good way for children to learn pattern recognition. According to Lee et al. (2014), children participating in CTArcade provided remarkably less pattern recognition examples than students participating in paper-based games. Sung and Black (2021) found that the embodied approach significantly improved students' pattern recognition skills. Saxena et al. (2020) used LEGO pattern as a hands-on pattern-building activity for students to learn pattern recognition. The pattern recognition skill was also observed in an unplugged engineering design activity, especially in the process of idea generation, idea representation and design evaluation (Ehsan et al., 2021).

2.3.5.3 Generalizing

Generalizing is the ability to transfer a specific problem-solving strategy into a different context (Qu & Fok, 2021). (Del Olmo-Muñoz et al., 2020) assessed children's generalization skills following the Bebras learning model. Qu and Fok (2021) assessed children's generalizing skills according to three levels (emerging, moderate, substantive) and found that student-robot interactions significantly improved children's generalizing skills.

2.3.5.4 Logical thinking

Logical thinking refers to the ability to arrange and analyze data (International Society for Technology in Education, 2011). Students who are good at logical thinking were more likely to succeed in CT activities, for example, by using terms such as "because … so …" or "if … then …" to express their ideas (Qu & Fok, 2021). Qu and Fok (2021) indicated that three types of S-R interaction (Programming-computing, Observational investigation, and Participatory investigation) might all involve children's logical thinking skills. Critten et al. (2022) found that children could develop logical thinking skills through guided play activities.

2.3.5.5 Simulation

Simulation refers to developing a (computational) model to imitate real-world processes (Dasgupta et al., 2017). In the study by Ehsan et al. (2021), the child became a model, imitating the natural process of a dog playing in a puppy playground to detect and debug problems.

2.3.5.6 Spatial reasoning

Although spatial reasoning is not a component of most CT frameworks, Clarke-Midura et al. (2021) incorporated it into the CT framework. They stated that many newly emerging tools and educational toys used for kindergarten CT instruction entail navigating an agent through two-dimensional grid space, which involves spatial reasoning (Clarke-Midura et al., 2021).

2.3.6 Classic CT perspectives

Among the reviewed studies, 12 (28.57%) were designed to enhance children's expressing perspective and 15 (35.71%) to enhance connecting perspective. The questioning perspective is not involved in the reviewed studies.

2.3.6.1 Expressing

For a computational thinker, computation is not just about consumption but also a means of creating and self-expression (Brennan & Resnick, 2012). Researchers have developed different tools and curricula to support children's creative expression. For example, the wooden platforms with the KIBO robotics kits are designed to facilitate content creation (Bers et al., 2019; Elkin et al., 2016; Sullivan & Bers, 2018). ScratchJr allows kids to create animations, collages, stories, and games (Portelance et al., 2016). The CAL-KIBO curriculum taught programming as a symbolic representation system for expression and creativity rather than problem-solving (Relkin et al., 2021). Children were usually encouraged to create at the end of the course, which allowed them to apply programming concepts they learned earlier toward a personally meaningful project (Bers et al., 2019; Portelance et al., 2016; Strawhacker & Bers, 2015; Sullivan & Bers, 2018).

2.3.6.2 Connecting

Brennan and Resnick (2012) indicated two ways of connecting with others. One is creating with others through communication and collaboration, which makes children do more than they could have on their own. Another way is creating for others, i.e., sharing the work they have created with others. Bers et al. (2014) found that this could promote children's motivation.

Researchers suggested various approaches to foster children's connecting perspective. In studies by Bers et al. (2019) and Pugnali et al. (2017), students attended a Technology Circle at the end of each activity, which enabled them to communicate and share their ideas. In Sullivan and Bers's (2018) study, the school community was invited to attend a presentation made by children, which allowed children to share works with others. Murcia and Tang (2019) demonstrated that children displayed positive emotional and social outcomes from jointly constructing a computational product or solving a problem. Critten et

al. (2022) showed that even two years old children could communicate and collaborate through guided play activities.

2.3.7 Emerging CT perspectives

Perseverance (2 papers, 4.76%) and choices of conduct (4 papers, 9.52%), which emerged in our review, were not included in the three-dimensional CT framework.

2.3.7.1 Perseverance

Perseverance in the face of difficulty is critical to children's CT learning and problem-solving skills development (Sullivan & Bers, 2018; Wang et al., 2020). In Wang et al.'s (2020) study, the exemplary teacher inspired children to persist by using the intimate relationships between children and the robot, modeling an attitude of treating errors as part of the problem-solving process and giving positive feedback on children's small-steps progress. In Sullivan and Bers's (2018) study, the teacher interviews and reflective journals indicated that students could persevere in the face of challenges, which is preliminary proof that robotics helps students form the "can-do spirit needed in innovation".

2.3.7.2 Choices of conduct

Choices of conduct refer to the ability of self-regulating and making conscious decisions about one's behavior (Pugnali et al., 2017). Examples of choices of conduct such as following classroom rules and using materials responsibly (Sullivan & Bers, 2018). Three studies evaluated children's choices of conduct by using the Positive Technological Development Engagement Checklist (Bers et al., 2019; Pugnali et al., 2017; Sullivan & Bers, 2018). Nevertheless, none of the studies indicated whether their curricula significantly influenced children's choices of conduct.

2.4 Discussion

The main findings of our systematic review are: (1) The existing literature in ECE reported different components of CT unevenly; (2) While the three-dimensional CT

framework covers some of the CT components involved in ECE studies, there are some emerging components that are essential for young children; (3) Based on the studies we examined, some classic components of the three dimensional CT framework might not be appropriate for young children. According to these findings, the three-dimensional CT framework needs to be refined to construct the CT curriculum framework for ECE.

2.4.1 The CT curriculum framework for ECE: combining classic and emerging components

To develop a CT curriculum framework for ECE, we retained the CT components in the three-dimensional CT framework which were proven appropriate for children, incorporated the emerging CT components, and removed the CT components that were inappropriate for young children. For CT concepts, we retained sequences, loops, events, and conditionals in our CT curriculum framework, for they were proved age-appropriate for young children by empirical studies. We removed operators and data since these were not the primary component of CT in ECE and no empirical studies examined them. Meanwhile, we incorporated representation, control flow/ structures, and hardware/software for they emerged in the empirical studies and were proved developmentally appropriate for young children. For CT practices, we retained testing and debugging, being iterative and incremental/(engineering) design process, abstraction, modularizing/decomposition/problem reformulation in our CT curriculum framework for empirical studies proved their age appropriateness for young children. We removed reusing and remixing for no empirical studies explored these components in the context of ECE. Instead, we incorporated algorithms, pattern recognition, and generalizing, for they emerged in the empirical studies and were proved age-appropriate for young children. For CT perspectives, we retained connecting and expressing in our CT curriculum framework since empirical studies showed that they are age-appropriate for young children. The component of questioning was excluded

due to its irrelevance and a lack of empirical studies examining it. We incorporated perseverance and choices of conduct because they were identified in empirical studies and were found to be age-appropriate for children.

It should be noted that we have combined some related components or slightly modified several concepts in our CT curriculum framework. First, sequence, loops, events, and conditionals were subsumed under control flow/structures since researchers indicated that these concepts are sub-concepts of control flow/structures (Bers, 2018; Bers et al., 2014; Sullivan & Bers, 2013). Second, we grouped "problem reformulation" and "decomposition" into one, as they express the same idea (Wang et al., 2020), and used "decomposition" in our CT curriculum framework since decomposition arose most often. Third, we combined "being iterative and incremental" and "engineer design process" into one, as they all refer to a continual process of improving the work to achieve the optimal solution or product. And we used "iteration" to refer to these two concepts in our CT curriculum framework. Lastly, we adjusted "expressing" to "expressing and creating". Although expressing has the idea of creating in Brennan and Resnick's (2012) framework, we use "expressing and creating" to highlight the component of creating in our CT curriculum framework.

In addition, we excluded some CT components, although they have been examined in the empirical studies. These components include (1) Logical thinking: although two studies (i.e., Critten et al., 2022; Qu & Fok, 2021) used logical thinking as a component of CT, it is a broad term and often incorporates the concepts of abstraction and decomposition (Selby & Woollard, 2013); and (2) Parallelism, spatial reasoning, simulation, automation: among the reviewed studies, only Gordon et al.'s (2015) study briefly mentioned parallelism, Clarke-Midura et al.'s (2021) study involved spatial reasoning, Ehsan et al.'s (2021) study involved simulation, and Khoo's (2020) study involved automation. Due to the underrepresentation of these four components, we did not include them in our CT curriculum framework.

We combined classic and emerging components to develop the CT curriculum framework for ECE through this systematic review, as presented in Figure 3. Detailed descriptions of the CT curriculum framework can be found in Table 3.

Figure 3 *CT curriculum framework for ECE: A three-dimensional model*

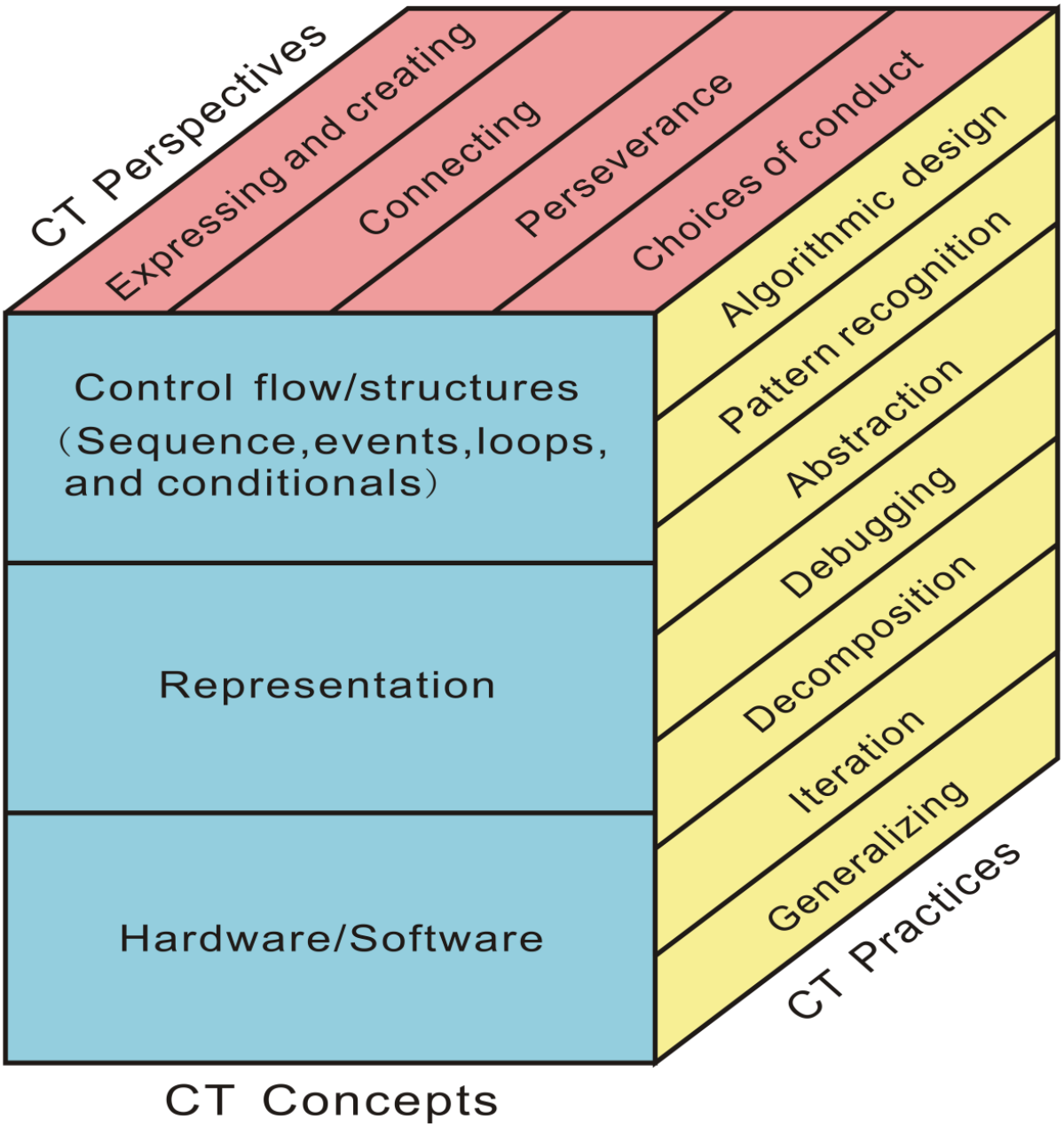


Table 3 *The Early Childhood CT Framework*

CT concepts	Description	CT practices	Description	CT perspectives	Description
Control flow/structures	The sequence in which instructions/commands are followed and executed (Bers, 2018) Control flow/structures in ECE include sequence, loops, events, and conditionals.	***Algorithmic thinking***	Designing a series of ordered commands to accomplish a task or reach a goal effectively (Bers, 2018)	***Expressing and creating***	Treating computation as a way to create and express ideas (Brennan & Resnick, 2012)
		Pattern recognition	Finding patterns or similar characteristics to simplify the solution (Hsu et al., 2018)	***Connecting***	Communicating and cooperating with others to accomplish a task or solve a problem together, and sharing works with others to get feedback (Brennan & Resnick, 2012)
		Abstraction	Exclude unnecessary or unneeded details when solving a problem (Lee et al., 2022)		
Representation	Symbols can represent	***Debugging***	Finding and fixing errors when solutions failed to function as expected (Wang et	***Perseverance***	Being persistent when encountering difficulties or

	concepts, actions, sounds, and more (Bers, 2018)		al., 2020)		failures, and treating failures as a natural process of achieving a goal (Wang et al., 2020)
		Decomposition	Breaking down a complex problem or system into smaller, easier-to-manage pieces (Wing, 2011)		
	The hardware follows the instructions set in the software to accomplish tasks as a system (Bers, 2018)	***Iteration***	Repeating the design process to seek improvements until the ideal solution is found (Shute et al., 2017)	***Choices of conduct***	Conscious decision-making about one's behavior (Pugnali et al., 2017)
Hardware/ Software		***Generalizing***	Transferring solutions used to solve specific problems to new contexts (ISTE, 2011)		

2.4.2 Limitations of the systematic review and the CT curriculum framework

We construct a CT curriculum framework for ECE based on the systematic review and the three-dimensional framework proposed by Brennan and Resnick (2012). Therefore, it is supposed to be clearly structured and relatively comprehensive. However, this framework may also have some possible omissions because it only focuses on Brennan and Resnick's (2012) framework and does not compare with other CT frameworks. In the future, we can compare the similarities and differences between different CT frameworks for K-12 education to verify the comprehensiveness of the early childhood CT curriculum framework and improve it further.

In addition, understanding CT learning trajectories is critical to improving the implementation and effectiveness of CT education. However, this framework does not specify which concepts, practices and perspectives children of different ages (between 2 and 8 years old) should learn and what developmental level they can achieve. Therefore, future research should focus on learning trajectories in CT to help practitioners understand young children's learning and developmental characteristics in CT.

2.4.3 Implications for research, policy, and practice

The refined CT curriculum framework for ECE is of great theoretical and practical importance for research, policy, and practice.

First, this robust framework clarifies what components should be included in the CT curriculum framework for ECE, thus making a significant theoretical contribution. It can also facilitate subsequent investigations to be conducted within a unified CT curriculum framework for ECE.

Second, this framework provides an essential reference for policymakers in developing a guideline for CT education in early childhood. Although CT is recognized as an essential skill for the 21st century, it is not currently included in the policy documents for

ECE in many regions/countries. Many CT education programs for young children are often provided as after-school programs (Ahn et al., 2021; Sung & Black, 2021) or summer camp programs (Pila et al., 2019; Pugnali et al., 2017; Qu & Fok, 2021). One of the key reasons is that CT education in early childhood is still not yet supported by educational authorities, especially when there is not yet a unified CT curriculum framework for ECE. The refined CT curriculum framework will support the development of the policy guidelines and promote the development and dissemination of CT education in early childhood settings.

Third, this framework can guide teacher educators and professional development providers to train teachers to integrate CT education into their classrooms. Strawhacker et al. (2018) found that teachers with more content knowledge could facilitate children's CT learning more effectively. However, preschool teachers lack the content knowledge to support children's learning of CT (Strawhacker et al., 2018; Wang et al., 2020). The proposed CT curriculum framework for ECE enables teacher educators to provide teachers with a comprehensive understanding of the content of CT education in early childhood settings.

Chapter 3: Teaching Programming and Computational Thinking in Early Childhood Education: A Case Study of Content Knowledge and Pedagogical Knowledge

Yue Zeng[1 2], Weipeng Yang[2], and Alfredo Bautista[2]

[1] School of Education, Wenzhou University, Wenzhou, China; Department of Early Childhood Education, The Education University of Hong Kong, Hong Kong SAR, China

[2] Department of Early Childhood Education, The Education University of Hong Kong, Hong Kong SAR, China

Abstract

Programming and computational thinking (CT) have been progressively incorporated into early childhood education to prepare children for the digital age. However, little is known about the content knowledge (CK) and pedagogical knowledge (PK) possessed by early childhood teachers in this domain. To address this gap, we conducted a case study of an early childhood teacher in China who had experience developing and implementing an unplugged programming and CT curriculum. The triangulation of data sources was established to collect evidence from videotaped observations, interviews, and lesson plans. For the CK, analysis of these findings revealed that the teacher had a more robust understanding of CT concepts (e.g., sequences, conditionals, and loops) compared to CT practices (e.g., decomposition, debugging) and CT perspectives (e.g., perseverance, choices of conduct). In terms of PK, the teacher could apply the general pedagogical knowledge but was relatively weak in using content-specific pedagogical knowledge. As the first endeavor to investigate an early childhood teacher's CK and PK in teaching programming and CT, this study provides significant implications for improving teachers' professional knowledge and teaching effectiveness in this burgeoning area.

Keywords: programming; computational thinking; early childhood teacher; content knowledge; pedagogical knowledge

3.1 Introduction

Globally, an increasing focus has been placed on teaching programming and computational thinking (CT) in early childhood education (ECE) (Bers et al., 2022; Yang et al., 2023). CT, viewed as a core competency in the 21st century, is related to solving problems that are often open-ended and complex in various disciplines with the use of the concepts fundamental to computer science (Wing, 2006). CT involves the ability to break down

complex problems into smaller parts, identify similarities among and within problems, develop step-by-step solutions and so on (Zeng et al., 2023a). Programming, on the other hand, is the process of writing codes to implement a particular task or solve a particular problem (Mills et al., 2021). CT and programming are closely intertwined, with each relying on and enhancing the other. Programming necessitates CT skills to create efficient and effective code (Lye & Koh, 2014), while programming plays a crucial role in the development of CT (Voogt et al., 2015). For example, when programming, a programmer often needs to break down a complex task into smaller parts, recognize patterns in data, and identify the most efficient approach for each step. This process involves CT skills such as pattern recognition, algorithmic thinking, and abstraction, which can then be applied to other domains, such as mathematics, science, and engineering.

Teachers' pedagogical content knowledge (PCK), which represents the incorporation of content and pedagogy into an understanding of how to make the teaching content easily understood by students with diverse abilities and interests (Shulman, 1987), is critical in predicting and enhancing young children's learning in domain-specific areas (Dunekacke & Barenthien, 2021). Previous research indicated that providing support for teachers' PCK had a positive impact on their teaching practices and children's development (Gözüm et al., 2022). However, few studies have examined early childhood teachers' PCK for teaching programming and CT. To fill this gap, this study aims to investigate early childhood teachers' content knowledge (CK) and pedagogical knowledge (PK) in teaching programming and CT. Specifically, we employed two frameworks to analyze an early childhood teacher's CK and PK that is demonstrated in planning, implementing, and reflecting on programming and CT activities. This investigation is crucial for providing training that focuses on addressing the areas of weak CK and PK among early childhood teachers, thus enhancing the effectiveness of teaching in early programming and CT.

3.1.1 Previous Studies on Unplugged Programming and CT Education

Programming and CT education is primarily conducted through two approaches: the plugged approach and the unplugged approach. The plugged approach involves using digital devices such as tablets, computers, and the Internet. In contrast, the unplugged approach aims to teach programming and CT without any digital devices, instead utilizing materials like pen and paper, cards, or engaging in physical activities (Otterborn et al., 2020).

Romero et al. (2018) summarized the key benefits of the unplugged approach, including embodied learning, reduced cognitive load, and concrete analogies. The unplugged approach often incorporates physical actions and tangible manipulation, aligning well with the learning styles of young children. Furthermore, compared to digital tools, incorporating unplugged materials in programming and CT education could minimize distractions that divert children's attention and reduce cognitive load, which refers to information-processing (attentional or working-memory) demands (Block et al., 2010). Lastly, unplugged activities are built upon the construction of tangible and concrete analogies, facilitating the learning of abstract concepts related to programming and CT. Several studies have explored the effectiveness of the unplugged approach in promoting learners' CT (Ahn et al., 2021; Del Olmo-Muñoz et al., 2020; Li & Yang, 2023; Saxena et al., 2020). In this study, the way the teacher employed to teach programming and CT is the unplugged approach.

3.1.2 The Content Framework of Computational Thinking in ECE

The goal of early programming and CT education is not to prepare children to become programmers or algorithmic engineers but rather to foster their CT. As argued by Resnick and Robinson (2017), children do not simply "Learn to Code" but rather "Code to Learn" and "Learn Through Coding". Thus, our interest lies in identifying the core content of CT covered and emphasized in early childhood teachers' instruction of programming and CT. To achieve this, we reviewed the CT content framework in ECE.

There is a lack of a consistent content framework for CT in ECE (Zhang & Nouri, 2019). After comparing different CT frameworks, Zeng et al. (2023a) used Brennan and Resnick's (2012) three-dimensional CT framework to identify CT components that were proven appropriate for young children to learn and established the CT curriculum framework for ECE. This framework articulates the core content in early programming and CT education, covers CT concepts (i.e., control flow/ structures, representation, and hardware/ software), CT practices (i.e., algorithmic design, pattern recognition, abstraction, debugging, decomposition, iteration, and generalizing), and CT perspectives (i.e., expressing and creating, connecting, perseverance, and choices of conduct) (Zeng et al., 2023a) (see Table 4).

Table 4 *The CT Content Knowledge Framework in ECE (Zeng et al., 2023a)*

CT dimensions	CT components	Descriptions
CT concepts	*Sequences*	A specific task or activity is conveyed as a succession of separate commands or steps that a human or machine can carry out (Brennan & Resnick, 2012)
	Loops	A mechanism of repeatedly executing the same instructions (Brennan & Resnick, 2012)
	Conditionals	Allowing for the expression of different outcomes by making decisions based on certain circumstances (Brennan & Resnick, 2012)
	Events	"One thing causing another thing to happen" (Brennan & Resnick, 2012, p. 4)
	Representation	In programming, representation refers to the use of symbols to represent instructions (Bers, 2018)
	Hardware/ Software	Hardware and software operate in tandem to complete tasks; the software gives the hardware instructions, and the hardware executes those instructions (Bers, 2018)
CT practices	*Algorithmic design*	A set of sequential, organized steps used to solve a problem or complete a task (Bers, 2018)
	Pattern recognition	Identifying patterns and trends (commonalities) between and within problems to simplify the solution (Hsu et al., 2018)
	Abstraction	The conscious effort to ignore irrelevant details and focus only on the important information, thus making problem solving easier (Lee et al., 2022)

	Debugging	Identifying and repairing mistakes when solutions do not work as expected (Wang et al., 2020)
	Decomposition	Breaking down a complex problem or system into smaller easily solved or managed parts (Wing, 2011)
	Iteration	Seeking upgrades of solutions using design processes repeatedly until the optimum solution is obtained (Shute et al., 2017)
	Generalizing	Transferring approaches used to address particular issues to new situations (CSTA & ISTE, 2011)
CT perspectives	*Expressing and creating*	Seeing computation as a way for designing and conveying ideas (Brennan & Resnick, 2012).
	Connecting	Cooperating, communicating with others and sharing works with others (Brennan & Resnick, 2012)
	Perseverance	Persevering in the face of challenges or failures and seeing failures as usual to reach a goal (Wang et al., 2020)
	Choices of conduct	Deciding what to do and what not to do in a specific situation by oneself (Pugnali et al., 2017)

3.1.3 Pedagogical Issues Related to Teaching Programming and CT in ECE

This section summarizes the teaching context, activity structure, pedagogical approaches, and pedagogical strategies previously used to foster children's programming and CT skills (see Table 5).

Table 5 *The Programming and CT Pedagogical Knowledge Framework in ECE*

Dimensions	Indicators	Description
Teaching context	*Group activity*	Purposeful, planned activities organized by the teacher in which many children in the class participate
	Learning center	Different learning areas in the classroom self-chosen and -directed by children
	Daily lives and routines	Children's daily lives and routines such as having meals, washing hands, and tidy up toys
	Integrative learning contexts	Connecting programming and CT with other learning domains such as art, math and literacy
Activity structure	*Highly structured*	Objectives pre-defined by teachers, and the activities primarily initiated by teachers
	Open-ended	Activities that allow children to freely explore
	Mixed	Activities that include both structured activities and open-ended activities and/or free play (Bakala et al., 2021)
Pedagogical approaches	*Task-based learning*	Teacher-directed pedagogical approach in which learning activities are organized around adult-guided tasks (McCormick & Hall, 2021)

	Project-based learning	Activities that allow children to explore relatively independently for long periods and yield real works or presentations (Kokotsaki et al., 2016)
	Problem-solving learning environment	A learning environment proposed by Lye and Koh (2014) that can enhance students' CT practices and perspectives, which include authentic problem, information processing, scaffolding and reflection
	Play-based learning	A playful, child-directed pedagogical approach with some adult direction and learning goals (Pyle & Danniels, 2017)
	Others	Other pedagogical approaches not covered in this list
Pedagogical strategies	*Unplugged activity*	Learning programming and CT without a computer and is often conducted through bodily activity or with other learning materials (Otterborn et al., 2020)
	Embodied cognition	Using embodied activities to help children understand abstract CT concepts (Moore et al., 2020; Saxena et al., 2020)
	External memory support scaffolding	Providing supplementary materials to turn abstract algorithms into visible and concrete representations to help children cope with working memory limitations and reduce cognitive load (Macrides et al., 2021)
	Pair programming	A collaborative programming approach in which two students work together to complete the same programming task (Denner et al., 2014)

Differentiated Instruction	Providing children with appropriate scaffolding based on each child's individual abilities and needs (Wang et al., 2020)
Demonstration	Modelling the necessary skills and attitudes to children (Wang et al., 2020)
Others	Other pedagogical strategies not covered in this list

3.1.3.1 Teaching Context

Lee and Junoh (2019) noted the importance of infusing programming and CT into children's daily lives and setting up programming centers/corners in early childhood classrooms. Mills et al. (2021) emphasized that integrating programming and CT into other learning domains would provide meaningful learning contexts for young children.

3.1.3.2 Activity Structure

There are three categories of programming and CT activity structure: *highly structured, mixed, and open-ended*. Most studies designed highly structured programming and CT activities (Khoo, 2020; Nam et al., 2019) and few studies designed open-ended free play with programming tools. Newhouse et al. (2017) found that the children appeared more engaged and motivated in the high teacher-supported sessions rather than in free play without explicit scaffolding. Other studies designed mixed activities (Bers et al., 2014; Bers et al., 2019). For instance, in the study by Strawhacker and Bers (2015), there was always a "buffer lesson" for children to explore the programming materials freely, which allowed them to absorb what they had learned and kept their attention throughout other highly structured activities.

3.1.3.3 Pedagogical Approaches

Early programming and CT education employs a variety of pedagogical approaches. One such approach is the task-based approach, where learning activities revolve around tasks

guided by adults (McCormick & Hall, 2021). Bers (2019) showed how such intentionally structured activities can aid young children in developing CT skills. Another notable approach is the project-based learning, characterized by its student-centered nature. This approach emphasizes students' autonomy, goal-setting, planning, exploration, cooperation, and reflection within authentic real-world practices (Kokotsaki et al., 2016). Several studies involved activities of the construction of robots, engaging students in design, problem-solving, decision-making, and investigative tasks (Macrides et al., 2021). Play-based learning, on the other hand, presents a playful and child-directed pedagogical approach with some adult guidance and predefined learning objectives (Pyle & Danniels, 2017). Critten et al. (2022) suggested play-based, pedagogic practices can be used with children as young as 2 years to learn many of the basic concepts involved in CT skills. Moreover, Lye and Koh (2014) suggested designing a problem-solving learning environment, which includes authentic problems, information processing, scaffolding and reflection, to enhance students' CT practices and perspectives.

3.1.3.4 Pedagogical Strategies

Previous studies have examined the effectiveness of different pedagogical strategies for improving young children's CT, including unplugged activities, embodied cognition, external memory support scaffolding, and pair programming. Unplugged programming uses materials like paper, cards, and blocks and has been shown to improve CT skills through embodied learning, lower cognitive load, and concrete analogies (Otterborn et al., 2020; Romero et al., 2018). While for embodied cognition, there are two kinds of embodiment according to the source of body movement: direct embodiment, which refers to moving bodies to perform solution steps; and surrogate embodiment, which refers to manipulating an external surrogate without engaging their bodies (Fadjo, 2012b). External memory support scaffolding is used to help children cope with working memory limitations and reduce

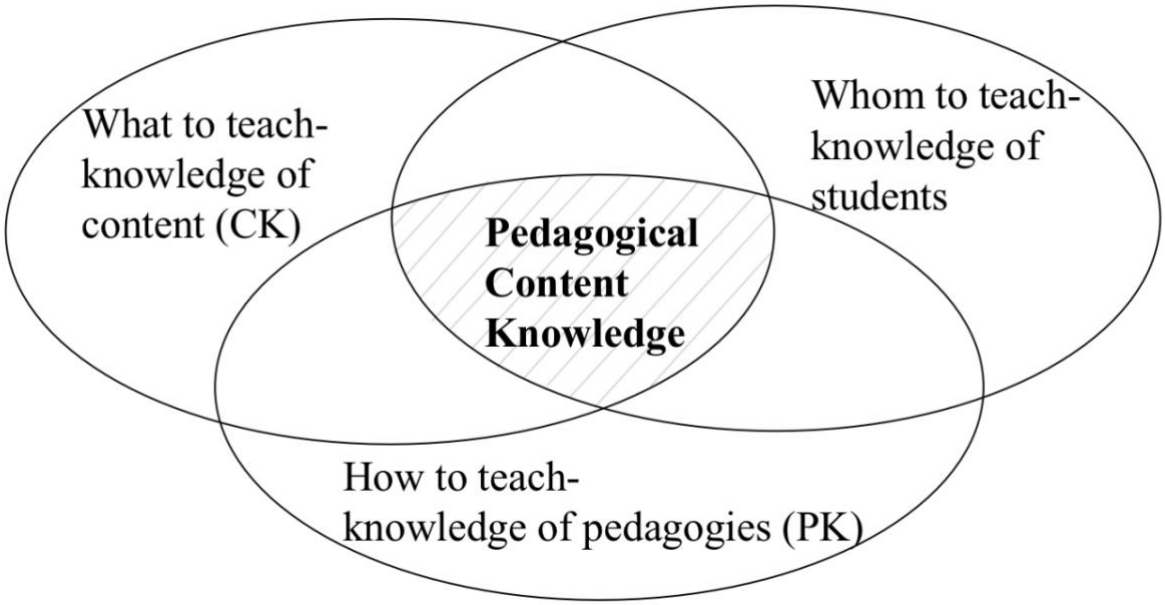
cognitive load during programming (Angeli & Valanides, 2020). Pair programming, a collaborative programming approach in which two students work together on a single computer to complete the same programming task, positively improved students' programming and CT skills, learning motivation, metacognition, and collaboration (Denner et al., 2014; Papadakis, 2018). Besides these experimental studies, Wang et al. (2020) video observed various strategies an exemplary teacher used to support preschoolers' CT skills, such as modelling a positive attitude toward error, breaking down problems into small steps, and providing different scaffolds according to children's individual needs.

However, previous studies were mainly aimed at validating the effectiveness of a particular pedagogical strategy in improving children's CT without examining what pedagogical strategies teachers used. Only Wang et al. (2020) investigated the pedagogical strategies used by a male teacher; however, this case study was conducted in a higher teacher-student ratio (1:3) context instead of a large-group context which is common in Asian cultural contexts.

3.1.4 The PCK Theory

PCK was first introduced by Shulman to emphasize the fundamental role of subject matter in (research in) teacher education and teaching in 1985. In subsequent years, PCK has been defined by different researchers in multiple ways. Despite the various definitions, researchers have identified three essential components of PCK: CK, PK, and knowledge of students' understanding (McCray & Chen, 2012; Rojas, 2008; Zhang, 2015). Figure 4 illustrates how these three components are interrelated to the construct of PCK (McCray & Chen, 2012). This study specifically examined teachers' CK and PK of programming and CT.

Figure 4 *Pedagogical Content Knowledge (PCK) (McCray & Chen, 2012, redrawn)*



CK is the knowledge of what to teach. It encompasses knowledge of the discipline to be taught, a thorough understanding of that knowledge, and an understanding of the relationships between topics of the discipline (Krauss et al., 2008). In this study, we focused specifically on the first two aspects, i.e., whether the teachers knew the programming and CT knowledge to be taught and whether teachers had a deep understanding of them.

PK is the knowledge of how to teach. There are two types of PK: general pedagogical knowledge (GPK) and content-specific pedagogical knowledge (CPK). GPK comprises comprehension of various educational philosophies and learning theories, general knowledge of learners and basic teaching rules, and familiarity with classroom management principles and strategies (Grossman & Richert, 1988). CPK is the knowledge of instructional strategies unique to a particular subject or topic (Zhang, 2015). In this study, we examined both the GPK and CPK.

3.1.5 Teachers' PCK of Programming and CT

Given the scant existing literature in this field, we conducted a comprehensive review focusing on the PCK of both preservice and in-service teachers across all educational levels.

Several researchers have discovered that both pre-service and in-service teachers possess limited knowledge of CT and little knowledge of how to teach programming and CT (Bower & Falkner, 2015; Chalmers, 2018; Sands et al., 2018).

Accordingly, it has been suggested by researchers that there is a pressing need to enhance teachers' PCK through pre-service and in-service training programs to facilitate the integration of CT into their classrooms (Chalmers, 2018; Haines et al., 2019; Yadav et al., 2017). Chalmers (2018) specifically emphasized that a deeper understanding of CT concepts, practices, and perspectives is crucial for teachers to effectively incorporate CT into the primary curriculum. Çakıroğlu and Kiliç (2020) proposed a course model and evaluation tools aimed at improving teachers' PCK for teaching CT via robotic programming.

Within the context of ECE, Strawhacker et al. (2018) found that teachers who possessed a solid foundation of CK exhibited more purposeful use of the programming tool and gave more explicit support. Similarly, Wang et al. (2020) found that the case teacher intentionally employed various strategies in his programming and CT instruction because of his clear understanding of CT skills that young children need to develop.

3.1.6 The Present Study

Previous research has indicated a need to improve teachers' PCK through training to help them implement programming and CT education (Chalmers, 2018; Haines et al., 2019; Yadav et al., 2017). To provide targeted training to help teachers acquire the necessary PCK and effectively deliver programming and CT education, it is crucial to clearly understand the status of teachers' PCK in programming and CT education. However, based on our thorough review of the existing literature, there is a lack of research specifically examining the status of CK and PK of programming and CT among early childhood teachers. As teachers' CK and PK can be demonstrated in their teaching (Zhang, 2015), to examine early childhood teachers' CK and PK, we proposed the following questions:

RQ1. What CT concepts, practices and perspectives were covered and emphasized in the early childhood teacher's teaching of programming and CT?

RQ2. How did the early childhood teacher support children's programming and CT learning?

3.2 Method

We employed a case study method, which allows people to gain a greater insight into a specific case by investigating it in depth and within its actual context (Yin, 2009). Our case study examined an early childhood teacher's CK and PK in teaching programming and CT.

3.2.1 The Research Site

This study was conducted in a provincial first-class public kindergarten located in Wenzhou, China. In 2022, when I conducted my research, few kindergartens in Wenzhou were involved in programming and CT education. Upon learning that this kindergarten had initiatives in this field, I contacted the principal and secured her informed consent.

This kindergarten has been committed to science education since 2004. With the introduction of STEM education, the school recognized its benefits in cultivating problem-solving skills and interdisciplinary competencies among young children. Consequently, they took the lead in implementing STEM curricula. In 2017, the kindergarten was selected as one of Wenzhou's first STEM pilot schools. During this period, the kindergarten collaborated with experts in the STEM field, who emphasized the importance of incorporating programming education in early childhood learning. Recognizing the increasing importance of programming and CT education, the kindergarten embarked on a new educational initiative to integrate programming and CT into its curriculum.

As an initial step, instead of implementing programming and CT education across all classes, the kindergarten decided to initiate a pilot program. They selected one class from each of the age groups: K1 (3-year-olds), K2 (4-year-olds), and K3 (5-year-olds), led by one

teacher in each class. We chose the K3 class for observation because the teaching content of the unplugged curriculum in the K3 class was built upon that of K1 and K2 and covered all the CT skills of the unplugged curriculum, thus allowing us to examine RQ1 comprehensively. The K3 class consisted of 32 children aged 5-6 years, along with two teachers and a nurse. For each class, only one teacher was assigned to conduct the programming activities, with the other teacher and nurse not involved in implementing the programming curriculum. In the K3 class, Ms. Wu is tasked with implementing the programming curriculum. The 32 children in the class are split into two groups for programming activities. While Ms. Wu instructed one group of children in programming in the classroom, the other group was led by the other teacher to a separate room for games. The nurse's primary duties involve maintaining classroom cleanliness, ensuring children's safety, and aiding the teacher in preparing materials for activities. That is to say, the other teacher and nurse played a supportive role in programming education. For the purposes of this study, we selected Ms. Wu, who was responsible for teaching programming and CT in this class and who enthusiastically volunteered to join our study.

Initially, the three experimental classes utilized a plugged-in programming tool named MOBLO. MOBLO is a hybrid kit that enables young children to program a virtual character on the screen by manipulating tangible programming blocks. While children enjoyed MOBLO, parents expressed concerns about screen time and its potential impact on their children's eyesight. In 2020, the kindergarten applied for funding to develop unplugged programming tools and courses. Upon receiving the grant, they created unplugged programming tools based on the existing plugged programming tools. By 2021, the kindergarten officially implemented unplugged programming courses, offering a screen-free alternative for young learners to explore programming.The three experimental classes conducted the unplugged programming curriculum. Notably, Ms. Wu not only implemented

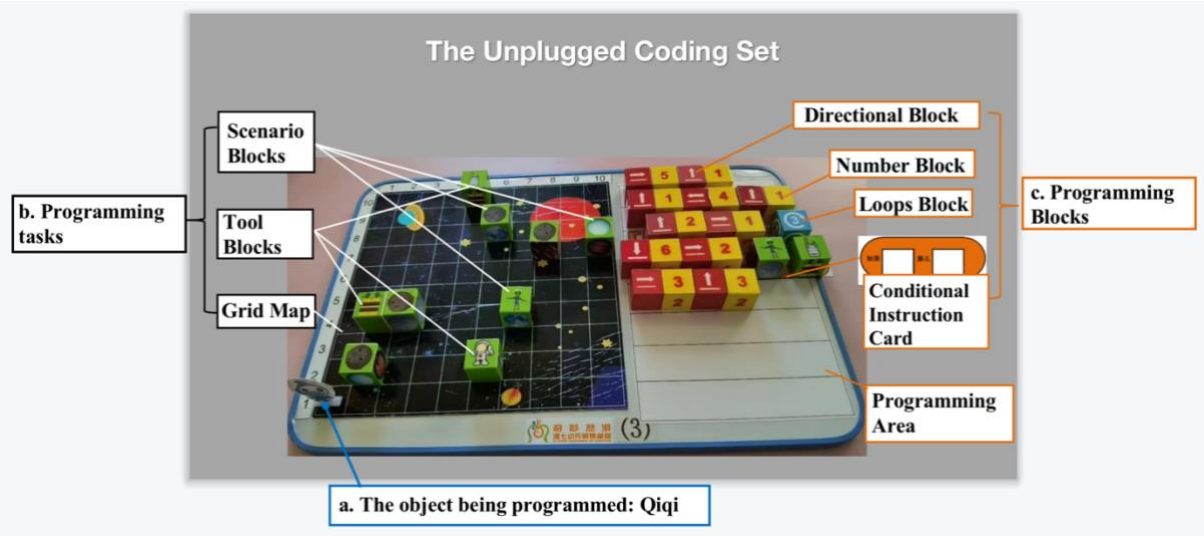
the unplugged curriculum but also participated in designing the unplugged programming tool and the unplugged curriculum.

Ms. Wu, a female teacher, possessed 11 years of work experience in the field of ECE. She initially graduated from a local normal university with an Associate's degree in ECE. Following 7 years of work experience, she pursued a Bachelor's degree in ECE through adult correspondence education. However, neither of these programs included any courses related to early programming or CT education. Ms. Wu's exposure to programming education came exclusively from the MOBLO company. To implement programming education with the MOBLO programming tool in the kindergarten, the MOBLO company provided training to teachers. This training primarily focused on instructing teachers on the utilization of the MOBLO programming tool and how to teach programming using the lesson plans provided by the MOBLO company.

The unplugged programming tool developed by this kindergarten, like other coding sets, consists of three parts: (1) The object being programmed: The object being programmed in this coding set is a pawn named Qiqi, who is also the protagonist of the stories in the unplugged curriculum; (2) Programming tasks: The teacher or children set up programming tasks by putting the Tool Blocks (representing tools Qiqi needs to obtain to solve problems) and Scenario Blocks (representing the characters, place, and things that happen in the story) on a 10 by 10 Grid Map. (3) Programming blocks: Children program the routes Qiqi takes by placing wooden Programming Blocks (including Directional Blocks, Number Blocks, Loops Block, and Conditional Instruction Card) in the Programming Area. For example, in the context of exploring outer space, Qiqi first needs to collect tools such as the spacesuit, oxygen kit, and translator. On his journey to other planets, he must avoid meteorites. When encountering problems, he needs to use tools (for example, using a translator when meeting an alien). Eventually, he reaches other planets (see Figure 5). Appendix 3 shows how to make

a similar coding set using readily available materials.

Figure 5 *The Unplugged Coding Set*



3.2.2 Data Collection

As teachers' PCK can be demonstrated in planning, implementing and reflecting on teaching (Zhang, 2015), lesson plans, videotaped programming activities, and audiotaped interviews were collected as our data to establish triangulation (Yin, 2009), as well as memos following each observation and interview.

3.2.2.1 Video Observations

Compared to other data types, video has definite advantages in capturing the teaching content and pedagogies in classrooms (Jacobs et al., 1999). In conducting the video observation, two cameras were used, one was set in the corner of the classroom to ensure the whole class activities were recorded, and the other was held by the researcher to capture Ms. Wu's interaction with children. A total of 12 lessons, each lasting approximately 40 minutes, over 6 weeks were video recorded, resulting in 728 minutes of video.

3.2.2.2 Interviews

We developed an interview protocol that focused mainly on two themes (in addition to a set of background questions): (1) Content Knowledge: Core content covered in the

programming and CT course and the early childhood teacher's understanding of them. (2) Pedagogical Knowledge: Pedagogical practices about supporting children's programming and CT learning (including a focus on the teaching context, activity structure, pedagogical approaches, and pedagogical strategies), as well as the reasons for adopting these pedagogical practices.

We conducted both formal and informal interviews. The formal interview was conducted after all sessions to understand Ms. Wu's CK and PK in early programming and CT (the interview protocol, see Appendix 2). It lasted around an hour. Informal interviews were conducted after class (if necessary) to have a deeper understanding of what had been observed.

3.2.2.3 Lesson plans

This study used lesson plans to supplement the observational and interview data. We collected a total of 12 lesson plans from Ms. Wu.

3.2.3 Data Analysis

To analyze the CT concepts, practices and perspectives that are covered and emphasized in the early childhood teacher's teaching of programming and CT, we used the CT curriculum framework for ECE (Zeng et al., 2023a), which has a detailed and clear definition of each CT component, as the CK Framework (see Table 4). The CK Framework includes three dimensions: CT concepts, practices, and perspectives.

To examine how the early childhood teacher supported children's programming and CT learning, we developed the PK Framework (see Table 5). The PK Framework comprises four dimensions: teaching context, activity structure, pedagogical approaches, and pedagogical strategies. We constructed the indicators under each dimension based on the aforementioned literature review. Moreover, we provided a clear definition for each indicator in the PK framework (see Table 5).

Then the first author and the second author used the two frameworks to analyze the video data, the interview data, and the lesson plans. The following explains the process of data analysis. Appendix 1 shows a few examples of data analysis.

3.2.3.1 Video and Interview Data Analysis

We analyzed recorded videos and interviews with the following steps:

Step 1. Transcription of selective video clips and interviews.

We rewatched all videos and selected informative video clips that could reflect Ms. Wu's CK and PK. The first author transcribed the selective video clips manually on her own. Before embarking on transcription for this project, she was trained in classroom video transcription. She had already transcribed classroom videos sufficiently, demonstrating high precision in translating video into text. The recorded interviews were also transcribed with utmost care and precision.

Step 2. Review and labeling of relevant information.

We carefully reviewed the transcriptions of the videos and interviews and highlighted the text related to CK in yellow and underlined the text related to PK.

Step 3. Identification of CK and PK indicators.

According to the CK and PK frameworks, we identified the CK and PK indicators in the transcriptions. We examined the CK and PK present in several videos and segments of interviews to guarantee the reliability of CK and PK extraction. After reaching 90% accuracy, the first author identified the CK and PK indicators involved in the rest of the videos and interviews.

3.2.3.2 Lesson Plan Analysis

The lesson plans were used for analyzing the teacher's CK and PK. Together, we first read through the 12 lesson plans and labeled vital information related to the research questions. Collaboratively, we proceeded to identify the CK and PK indicators involved in

the 12 lesson plans according to the CK and PK frameworks.

3.2.4 Ethical and Validity Issues

This study was conducted with the ethical approval from the Human Research Ethics Committee (HREC), the authors' university (Reference No. 2021-2022-0334). A letter outlining the research and consent forms was provided to the kindergarten principal and Ms. Wu, and permission was obtained from them. Since the children in this study were 5-6 years old, letters and consent forms were also provided to parents/guardians through kindergarten.

The findings were validated through data triangulation, member checking and inquiry auditing (Creswell, 2014). We collected data from multiple sources for triangulation. Member checking was conducted by re-interviewing Ms. Wu to ensure that her ideas stayed in line with her previous interview responses and the researchers' interpretations. In addition, two senior researchers in ECE acted as the auditors to ensure the rigor of the research procedure and confirm that the findings appropriately reflected important aspects of the observations, interviews, and lesson plans.

3.3 Findings

3.3.1 CT Concepts, Practices, and Perspectives Taught by the Teacher

Our evidence revealed that Ms. Wu emphasized CT concepts across her programming and CT teaching instead of CT practices and CT perspectives (see Table 6).

Table 6 *Frequency of Each CT Skill in Different Data*

CT dimensions	CT components	Videos	Interviews	Lesson plans
CT concepts	Sequences	12 (100%)	12 (100%)	12 (100%)
	Loops	10 (83.3%)	10 (83.3%)	10 (83.3%)
	Conditionals	10 (83.3%)	10 (83.3%)	10 (83.3%)

	Events	12 (100%)	0	0
	Representation	12 (100%)	0	0
	Hardware/ Software	0	0	0
CT practices	Algorithmic design	12 (100%)	0	0
	Pattern recognition	10 (83.3%)	0	0
	Abstraction	0	0	0
	Debugging	0	0	0
	Decomposition	0	0	0
	Iteration	0	0	0
	Generalizing	0	0	0
CT perspectives	Expressing and creating	5 (41.7%)	5 (41.7%)	5 (41.7%)
	Connecting	12 (100%)	12 (100%)	12 (100%)
	Perseverance	3 (25%)	0	0
	Choices of conduct	0	0	0

Note a: The notation "12 (100%)" indicates that the CT skill was present in all 12 PCT activities with a frequency of 100%. Similarly, "10 (83.3%)" indicates that the CT skill was present in 10 activities with a frequency of 83.3%, and so on.

Note b: During the interview, the 12 unplugged programming lesson plans were presented to the teacher, who was asked to identify the core content covered in each activity. The frequency of each CT skill was then calculated based on the teacher's responses.

3.3.1.1 CT Concepts

Our analysis found that Ms. Wu primarily focused on teaching CT concepts, particularly sequences, loops, and conditionals. These concepts were systematically integrated into her lessons, with sequencing introduced in K1, conditionals and loops introduced in K2 and further developed in K3.

While explicit instruction in the concepts of representation and events was absent from her teaching practices and lesson plans, children learned them through activities such as using Programming Blocks to represent routes and experiencing the correspondence between actions and instructions. The concept of hardware/software was not covered due to the constraints imposed by the unplugged programming tool.

3.3.1.2 CT Practices

The video data analysis showed a clear focus on algorithmic design and pattern recognition in the teaching of programming and CT. Algorithmic design was manifested in the development of routes, while pattern recognition was observed in creating repeated routes. However, neither of these terms was explicitly referenced during the interviews nor present within the lesson plans.

The teaching of other CT practices, including debugging, decomposition, abstraction, iteration, and generalizing, was neither evident in Ms. Wu's teaching practices nor present in the lesson plans. An example involved Ms. Wu's observation of an erroneous program created by a child. Instead of guiding the child to observe and identify the error, Ms. Wu worked with the child to remove the programming blocks from the programming area and let the child recreate the programs. This approach missed the opportunity to teach debugging skills to the child. Another instance where an opportunity for teaching decomposition emerged was during the "Backward Reasoning Task." This task necessitated children to complete a path based on information in the programming area and grid map. Although the task provided an opportunity to teach decomposition (see Figure 6), Ms. Wu did not introduce this skill. Additionally, these CT practices were not mentioned by Ms. Wu in the interview. When asked about the core content of early programming and CT education, as well as what children can learn from tasks such as "Backward Reasoning Task," Ms. Wu did not reference these CT practices.

Figure 6 *Ms. Wu Presented the "Backward Reasoning Task" with PPT*

3.3.1.3. CT Perspectives

Ms. Wu displayed an awareness of cultivating children's CT perspectives of creating and connecting; however, she did not give these aspects prominence in her instructional practices. Five activities designed by Ms. Wu involved fostering children's creativity; however, these primarily centered on encouraging children to design various routes to enhance their creativity, without affording them opportunities to apply their programming and CT skills in creating projects or expressing ideas, which could better cultivate children's creativity. In fostering connections among children, Ms. Wu employed pair programming; nonetheless, during pair programming, her focus was primarily directed towards checking the accuracy of the children's devised routes, while aspects of observing and facilitating collaboration received limited attention.

Moreover, she did not intentionally develop the children's persistence and choices of conduct. Throughout the 12 sessions, Ms. Wu showed concern for children's persistence only on three occasions; and she did not mention persistence in her interview or lesson plans.

Furthermore, while Ms. Wu underscored the importance of cultivating positive behaviors among children, her approach primarily relied upon issuing directives and reminders, rather than empowering children to make autonomous decisions.

3.3.2 Pedagogies Employed by the Teacher

Ms. Wu employed group activities to teach programming and CT with highly structured, task-based activities (see Table 7). She integrated CT skills into tasks that gradually increased in difficulty and guided children to learn CT skills by completing these tasks. Ms. Wu provided ample time and support for the children's self-exploration and acted as a facilitator and collaborator rather than an authority figure.

Table 7 *The Pedagogical Steps of a Programming Activity*

Time	Steps	Purpose of each step
1-2 mins	The teacher begins by telling a story and setting up a scenario for the activity.	To pique children's curiosity and engage their interest
10 mins or so	The teacher introduces Task 1 and teaches the key concepts through its completion by the children.	To instruct the core CT skills to the entire group
30 mins or so	The children work in pairs to complete Task 2 and/or Task 3 and share their completed work with the class.	To allow adequate time for children to practice, assess their work, offer prompt feedback, and explore common difficulties during the sharing session
1-2 mins	The children tidy up the programming	To cultivate positive habits in

materials. children

Ms. Wu utilized a range of pedagogical strategies to support children's programming and CT learning. Our analysis identified eight strategies, five of which were effective: the unplugged approach, contextualization, embodied cognition, external memory support scaffolding, and a step-by-step strategy for teaching loops. Ms. Wu embraced the unplugged approach to teaching programming and CT, which involves screen-free and hands-on activities. Additionally, she employed contextualization to provide meaningful contexts for programming and CT learning, such as integrating loops learning into the narrative of aiding Qiqi in exploring planetary mysteries. Furthermore, she utilized embodied cognition. The children interacted with an external surrogate named Qiqi, manipulating it to navigate a grid map based on provided instructions. They also physically moved within the grid on the floor, following the given directives. External memory support scaffolding was another strategy Ms. Wu employed, such as using visible Programming Blocks for notating children's algorithms to support their thinking and problem-solving. Furthermore, she implemented a step-by-step teaching strategy, drawn from early mathematics, to effectively teach loops.

However, the strategies of demonstration, pair programming, and differentiated instruction were not effectively utilized. While Ms. Wu often used demonstrations to exhibit how to identify coordinates, verify routes, and organize materials, she did not model problem-solving skills like debugging and decomposition, nor did she exemplify cooperation and a positive attitude towards mistakes. In addition, Ms. Wu employed pair programming, wherein two children with neighboring school numbers collaborated on programming tasks. However, observations indicated that Ms. Wu did not intentionally and actively observe and facilitate children's collaboration. Consequently, pair programming proved ineffective, frequently resulting in a lack of genuine interaction and cooperation between the two children, or in some instances, one child would assume a dominant role while the other

remained disengaged. Furthermore, Ms. Wu implemented differentiated instruction after recognizing that less capable children struggled to keep up and remained less engaged during programming and CT activities. However, her approach simply involved segregating children into two groups based on their abilities, slowing down the teaching pace, removing challenging tasks for the less capable group, and failing to provide targeted scaffolding for these children to grasp programming and CT concepts.

3.4 Discussion

This study represents a groundbreaking endeavor to investigate an early childhood teacher's CK and PK in teaching young children programming and CT. Video analysis revealed that Ms. Wu did the most intentional and systemic teaching in CT concepts. However, it was observed that she missed opportunities to expose children to CT practices (e.g., decomposition, debugging) and CT perspectives (e.g., perseverance, choices of conduct). This finding suggests that Ms. Wu possessed a robust foundation of knowledge regarding CT concepts but had limited knowledge of CT practices and perspectives. This conclusion was also supported by evidence from the interviews and lesson plans. Due to Ms. Wu's lack of clarity regarding the core CT practices and perspectives that children should learn, she did not intentionally integrate CT practices and perspectives into her teaching. As stated by Zhang (2015), if teachers lack an understanding of the diverse CK that should be taught, they will not devote sufficient time and effort to support children’s learning in certain areas. Notably, not only have CT practices and perspectives been neglected in educational practice, but there is also a lack of intervention studies on CT practices and perspectives. A literature review conducted by Lye and Koh (2014) on intervention studies revealed that the majority of studies (85%) solely focused on examining learning outcomes related to CT concepts, with only a small fraction (8 studies) reporting on CT practices or perspectives.

In terms of the learning context, researchers indicated that programming and CT are

everywhere in children's lives; integrating programming and CT into their daily routines and tasks, such as brushing teeth, washing hands, or making objects with clay, offers meaningful learning contexts (Lee & Junoh, 2019; Mills et al., 2021). However, according to interviews, Ms. Wu solely taught programming and CT through whole-group activities and was unaware of the learning opportunities present in daily activities and other learning domains. This can be attributed to Ms. Wu's limited CK in CT practices and perspectives. As noted by Zhou et al. (2006), teachers who possess strong CK can effectively support children's learning in any context.

Regarding activity structure and pedagogical approaches, it was found that Ms. Wu created a highly structured and task-based approach by carefully preparing materials and planning activities. This approach enabled Ms. Wu to offer substantial support for children's programming and CT learning, keeping them engaged in the assigned tasks. The significance of teacher scaffolding in facilitating children's programming and CT learning has also been highlighted by Newhouse et al. (2017) and Wang et al. (2020). They emphasized that without teachers' guidance, students are prone to losing interest in programming activities and are unlikely to demonstrate any actions that could be associated with an understanding of "algorithms". However, it is worth noting that while this approach allows teachers to provide sufficient guidance and support for young children, it may not good for fostering their autonomy and creativity (Kokotsaki et al., 2016).

This study identified eight pedagogical strategies Ms. Wu employed to support children's programming and CT learning. Among these, five were notably effective, while three showed limited effectiveness. Further analysis suggests that Ms. Wu's effective utilization of these strategies stems from her consideration of children's general learning characteristics. The unplugged strategy and embodied cognition align with the hands-on learning style commonly observed in young children (Macrine & Fugate, 2022). Similarly,

the contextualization strategy hinges on the widely recognized principle that children learn best when presented within engaging or authentic contexts that capture their interest (Perin, 2011). Additionally, the application of external memory support scaffolding corresponds with the well-established understanding that young children possess limited working memory capacity (Macrides et al., 2021). Only the step-by-step strategy for teaching loops considers the developmental trajectory of children when learning loops; however, according to Ms. Wu, this strategy was transferred from the mathematical domain.

Regarding the less effectively utilized teaching strategies, we found that their successful implementation demands a solid grasp of CK or children's developmental knowledge within the programming and CT domain. Effective demonstration, for instance, necessitates that teachers possess a strong understanding of the content within the programming and CT domains. This understanding enables them to precisely determine what aspects to model for young children and where to place emphasis during the modeling process. Similarly, successful pair programming requires sensitivity to the core content of "connecting" and proactive observation and intervention to facilitate children's productive collaboration in programming and CT learning. Additionally, differentiated instruction relies on teachers' awareness of children's developmental trajectory in programming and CT to provide tailored scaffolding.

These findings indicated that Ms. Wu exhibited proficiency in applying general pedagogical knowledge (GPK) to programming and CT education but was weak in utilizing content-specific pedagogical knowledge (CPK), which necessitates a solid understanding of the CK in programming and CT education. This finding supported the PCK theory, which Shulman introduced to emphasize the fundamental role of subject matter in teaching (Shulman, 1986). Ball and McDiarmid (1989) also pointed out that teachers with a deeper understanding of the teaching content were more likely to use effective pedagogical strategies

to enhance students' understanding of the subject matter. Moreover, Strawhacker et al. (2018) and Wang et al. (2020) also found that teachers with a stronger CK were better equipped to provide explicit scaffolding in programming and CT education.

Furthermore, the kindergarten in this study used an unplugged approach to teaching programming and CT. They developed an unplugged coding set consisting of three components - the object being programmed, programming tasks, and programming blocks. This board game coding set allows children to learn programming and CT. It is simple to reproduce, as it can be made using basic materials by following the steps provided in Appendix 3.

3.5 Limitations and Implications

3.5.1 Limitations

Although this study is the first to examine an early childhood teacher's CK and PK for early programming and CT, it does come with certain limitations. Firstly, this study was based on a single teacher as a case study. While this chosen case has provided insights into the teaching of programming and CT within the context of Chinese early childhood education, caution should be exercised when generalizing the findings to broader contexts or other educators. Secondly, this study solely focused on the early childhood teachers' CK and PK of programming and CT, without investigating the teacher's knowledge of students, which is a crucial component of teachers' PCK. Future studies should also explore early childhood teachers' understanding of students' learning in programming and CT.

3.5.2 Practical Implications

This study has important implications for practice. CT encompasses more than just CT concepts; it also involves CT practices and perspectives (Brennan & Resnick, 2012). When introducing CT in ECE settings, the goal is not simply to teach young children to "Learn to Code" but rather to equip them with problem-solving skills and attitudes that can be

applied beyond programming (Lye & Koh, 2014). CT practices and perspectives are exactly related to problem-solving skills and attitudes. However, it was found that the case teacher's intentional teaching in CT practices and perspectives was limited, and her knowledge of CT practices and perspectives was weak. Thus, it is crucial to provide teachers with professional support to help them understand the goal of programming and CT education and to enhance their knowledge of CT practices and perspectives. This will help teachers move from a focus on teaching CT concepts to intentionally integrating CT practices and perspectives into their teaching practices.

Furthermore, Ms. Wu's teaching approach was limited to whole-group activities. She lacked awareness of providing opportunities for children to learn and apply programming and CT in their daily lives. By developing a clear goal for programming and CT education and acquiring a strong CK in CT practices and perspectives, teachers can become equipped with the awareness, knowledge, and ability to integrate programming and CT into children's daily lives. This inclusive approach ensures that programming and CT skills are accessible to all children, particularly those from disadvantaged backgrounds.

Additionally, there are various pedagogical approaches for teaching early programming and CT, ranging from a task-based approach, where learning activities are centered around tasks guided by adults (McCormick & Hall, 2021), to project-based learning, which emphasizes student-centeredness (Kokotsaki et al., 2016). However, Ms. Wu solely employed a highly structured task-based approach, which prioritized guidance but overlooked students' autonomy and creativity. Therefore, teachers should adopt a flexible way by combining different pedagogical approaches to teach programming and CT. This enables the provision of necessary guidance while also encouraging students' autonomy and creativity.

Lastly, based on the unplugged programming tool developed by the case kindergarten, a guide for crafting an unplugged coding set has been innovatively proposed. Programming

tools play a crucial role in implementing programming and CT education. This age-appropriate, board game-like coding set extends young children the opportunity to engage in programming and CT activities within both formal and informal settings. Additionally, this unplugged coding set boasts ease of reproduction, as it can be made following straightforward steps and utilizing readily available materials.

3.5.3 Research Implications

This study makes an important contribution to the research. The constructed PK framework for programming and CT, based on an extensive literature analysis, provides a useful tool for analyzing teachers' PK in teaching programming and CT. In addition, the study presents a model case showcasing the application of CK and PK frameworks to investigate teachers' CK and PK in early programming and CT education.

Moreover, this study found that teachers had limited CK in CT practices and perspectives and insufficient content-specific pedagogical knowledge (CPK). Therefore, further research could explore ways to enhance teachers' pedagogical content knowledge in programming and CT education through training programs. It is especially important to investigate how teachers can effectively apply the acquired CK and CPK to their own teaching context to facilitate the integration of programming and CT into classrooms. Previous studies have demonstrated that coaching is critical in facilitating the transfer of training content to teachers' specific teaching situations (Neuman & Cunningham, 2009). Hence, future researchers could explore transferring the coaching model to the Chinese cultural context to enhance teachers' intentional and effective teaching of programming and CT.

Chapter 4: Developing Young Children's Computational Thinking through Programming with a Hybrid Kit

Yue Zeng[1 2], Weipeng Yang[2], and Alfredo Bautista[2]

[1] School of Education, Wenzhou University, Wenzhou, China; Department of Early Childhood Education, The Education University of Hong Kong, Hong Kong SAR, China

[2] Department of Early Childhood Education, The Education University of Hong Kong, Hong Kong SAR, China

.

Abstract

Fostering young children's computational thinking (CT) has garnered global interest as it aligns with the cultivation of 21st-century skills. Previous studies have focused on physical, virtual, and hybrid kits with virtual programming blocks, but rarely explored the use of hybrid kits that combine virtual sprites and physical programming environments. We conducted a quasi-experimental study to investigate the effect of using a hybrid programming kit with which children can program the action of the virtual sprite by manipulating tangible programming blocks on young children's CT. Furthermore, we explored the characteristics of children's programming engagement and the instructional strategies employed by teachers through video analysis and interviews. The results showed that: (1) Children's CT in the experimental group significantly improved, compared to peers in the control group. (2) Children's programming behavior demonstrated a change from "action preceding thought" to "thought preceding action" and from "relying on trial-and-error" to "active debugging" with the support of teachers. (3) Teachers used multiple strategies to support young children's programming. These findings further indicate the importance of introducing programming in early childhood education and emphasize the critical role that teachers play in supporting young children's learning of programming.

Keywords: Computational thinking; programming education; hybrid kit; engagement in programming; instructional strategies; early childhood education

4.1 Introduction

Computational thinking (CT), viewed as a 21st-century skillset that future generations must develop (Zhang & Nouri, 2019), is the systematic analysis, exploration, and testing of solutions to problems (Wing, 2011). CT plays a vital role in fostering the development of skills like planning, critical thinking, and problem-solving (Li & Yang, 2023). Furthermore, it has a broad impact on children's learning across other disciplines (Wang et al., 2020).

Programming is writing code to perform a specific action on a software program, application, or computer (Mills et al., 2021; Zhang et al., 2023). Programming is an essential approach to fostering CT (Voogt et al., 2015). Countries worldwide have been undertaking various initiatives to introduce programming in early childhood education to develop children's CT (Sullivan & Bers, 2018).

CT and programming are closely intertwined, with each depending on and reinforcing the other (Zeng et al., 2023b). Programming requires the utilization of CT skills (e.g., decomposition, pattern recognition, algorithmic thinking, and abstraction) to develop efficient and effective code (Lye & Koh, 2014), while CT is predominantly cultivated through programming (Voogt et al., 2015).

The objective of early programming education extends beyond training children to become programmers; its primary goal is to foster their CT abilities. As emphasized by Resnick and Robinson (2017), the intention is not merely for children to "learn to code" but, more importantly, to "code to learn" and "learn through coding".

The development of "Early Childhood CT Curriculum Framework" (Zeng et al., 2023a) and TechCheck-K (Relkin & Bers, 2021) provide the foundations for investigating the effect of programming on young children's CT. The "Early Childhood CT Curriculum Framework" presents the content that should be taught in early programming education and

provides guidance for designing programming activities in this study. The TechCheck-K, which measures CT in 5-9-year-olds through tasks such as problem-solving, sequence, graph decomposition, pattern recognition, shortest path determination, and obstacle course maze, resolves the problem of requiring young children to have a foundation in programming for CT assessment (Relkin & Bers, 2021), thereby serving as a feasible way of assessing CT in young children.

Yu and Roque (2019) classified programming tools into physical, virtual, and hybrid kits. Physical kits consist of tangible components. Virtual kits are PC and/or mobile-device-based applications without tangible components. Hybrid kits combine both tangible and virtual parts and can further be divided into two subcategories: “kits with physical robot and graphical programming environment” and “kits with virtual sprites and tangible programming environment” (Yu & Roque, 2019, p. 23). Previous studies that investigating the effectiveness of programming on young children's CT have primarily employed physical kits, such as Bee-Bot (Angeli & Valanides, 2020), KIBO (Bers et al., 2019), and Code-a-pillar (Wang et al., 2020). Additionally, some studies have utilized virtual kits, such as ScratchJr (Strawhacker et al., 2018) and Code.Org (Çiftci & Bildiren, 2020). There has also been exploration of hybrid kits combining a physical robot with a graphical programming environment, such as LEGO WeDo (Elkin et al., 2014). However, no studies have yet examined the effectiveness of hybrid kits with virtual sprites and tangible programming environment in promoting CT in young children (Yu & Roque, 2019).

Furthermore, existing research primarily focused on quantitatively validating the effectiveness of programming education in fostering children's CT, with limited literature exploring children's programming process. Only Wang et al. (2020) noted that 3-4-year-olds often required multiple attempts to fix errors; Qu and Fok (2021) observed that some 7-9-

year-old students made progress in CT skills through critical analysis and careful decision-making in robotics activities, while others did not; Chevalier et al. (2022) found that 8-9-year-olds who received immediate feedback without guidance relied on trial-and-error without planning and critically evaluating the program's syntax before programming.

Additionally, previous research has highlighted the importance of teacher guidance in learning to code (Angeli & Valanides, 2020; Newhouse et al., 2017; Wang et al., 2020). Studies have supported the effectiveness of instructional strategies such as embodied cognition (Fadjo, 2012a), external memory support scaffolding (Angeli & Valanides, 2020), and pair programming (Papadakis, 2018) in promoting children's CT. However, these studies primarily used quantitative research to validate the effectiveness of specific strategies, with a limited investigation into the instructional strategies early childhood teachers employed to support children's programming and CT learning. Only Wang et al. (2020) used video recordings of programming sessions to analyze the scaffolding strategies used by a teacher to support young children's CT learning. These strategies included dialogic scaffolding, a combination of high- and low-support scaffoldings, adapting scaffoldings to individual needs, and tailoring scaffoldings for different CT components. However, this study was conducted in a small group setting with one exemplary teacher and three preschoolers.

To address these gaps, we focused on examining the following questions:

RQ1. To what extent did learning programming using the hybrid kit with a virtual sprite and tangible programming blocks affect the child participants' CT?

RQ2. What were the characteristics of engagement in programming displayed by young children?

RQ3. What instructional strategies did teachers use to support young children's programming?

4.2 Method

4.2.1 Research Design

This study employed a control-group pretest-posttest design to investigate the impact of programming with a hybrid kit on young children's CT. The experimental class participated in programming activities designed by the researchers. To provide optimal support for each child's learning, the teachers randomly assigned the children in the experimental class to two groups. Both groups of children engaged in programming activities for approximately 45 minutes per week, working collaboratively in pairs. In contrast, the control class was divided into two groups for regular teaching activities.

Moreover, to capture the characteristics of children's engagement in programming and gain a deeper insight into the specific effects of programming on young children's CT, we conducted a 12-week video recording of the programming processes of a randomly selected group of two children.

Furthermore, we used both video recording and semi-structured interviews to investigate the instructional strategies employed by the teachers during the intervention, as they have a significant influence on children's programming and CT learning (Wang et al., 2020).

4.2.2 Participants

Convenience sampling was employed in this study, with children aged 5-6 from two classes in a public kindergarten in Wenzhou (China) being selected as the research participants. One class was randomly assigned as the experimental class, while the other class served as the control class. Each class had 35 children (18 boys and 17 girls) with similar mean ages (experimental: 6.16 years, SD = 0.32; control: 6.07 years, SD = 0.35). The majority of children (94.3%) had no prior programming experience.

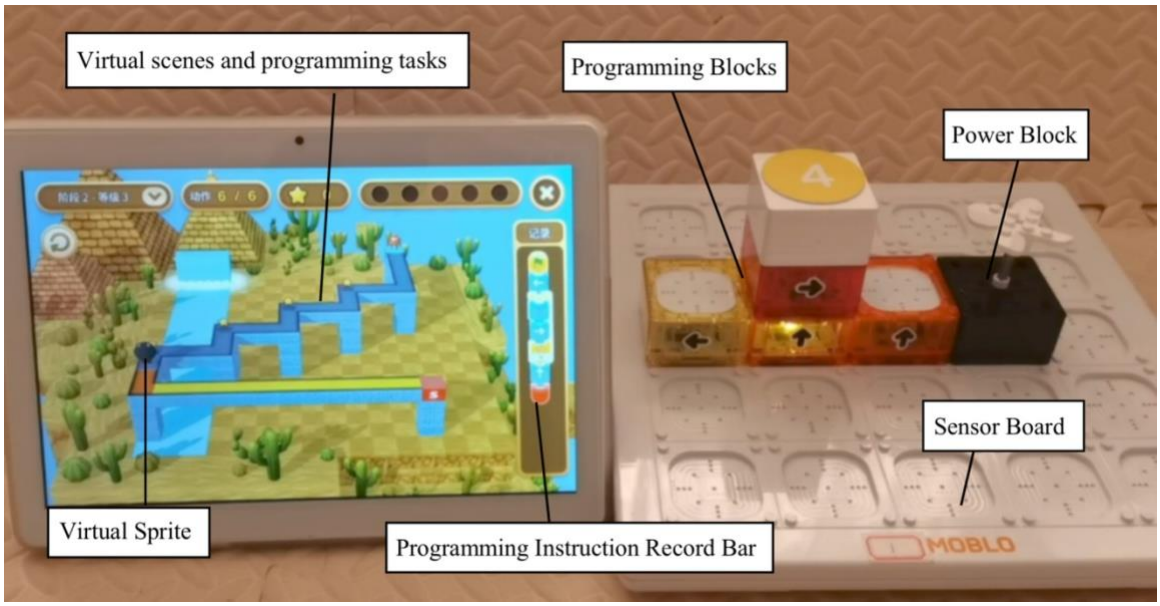
Two teachers from the experimental class participated in the study, each responsible for teaching programming to a group. Both teachers held a bachelor's degree in early childhood education and were certified as Kindergarten Level 1 teachers. They possessed 6 and 7 years of teaching experience in early childhood education, respectively. Neither had received any professional programming/ CT education training before participating in this study.

4.2.3 The Intervention

4.2.3.1 The Programming Tool

The programming tool used in this study was MOBLO developed by a Chinese company called *Mobaole*, which consisted of two parts: the tangible programming environment and the virtual programming game (see Figure 7). Children can program the action of the virtual sprite by manipulating tangible programming blocks.

Figure 7 *The MOBLO Programming Kits*



The tangible programming environment included (1) a Sensor Board for inputting programs; (2) various programming blocks for creating programs; (3) a Power Block for starting programs. The virtual programming game part included (1) a range of pre-designed

programming tasks; (2) a virtual sprite that executed the instructions children created; (3) a programming instruction record bar that displayed the instructions children created in real-time in symbolic form.

This programming tool provides diverse feedback for children's learning, both from the physical programming environment and the game interface. Feedback from the tangible programming environment is that as the virtual sprite executes the program, the corresponding electronic programming blocks illuminate with each step the virtual sprite takes, enabling children to visually connect the virtual sprite's actions with their created program. Feedback from the game interface mainly includes (1) the virtual sprite's actions can indicate the correctness of the program. If the program is right, the virtual sprite successfully reaches the destination and completes the task; in cases where errors exist in the program, the virtual sprite halts at the point where the program is wrong. (2) The programming instruction record bar presents the instructions entered by the child in real-time, which helps the child associate the tangible programming blocks with the symbolic representations.

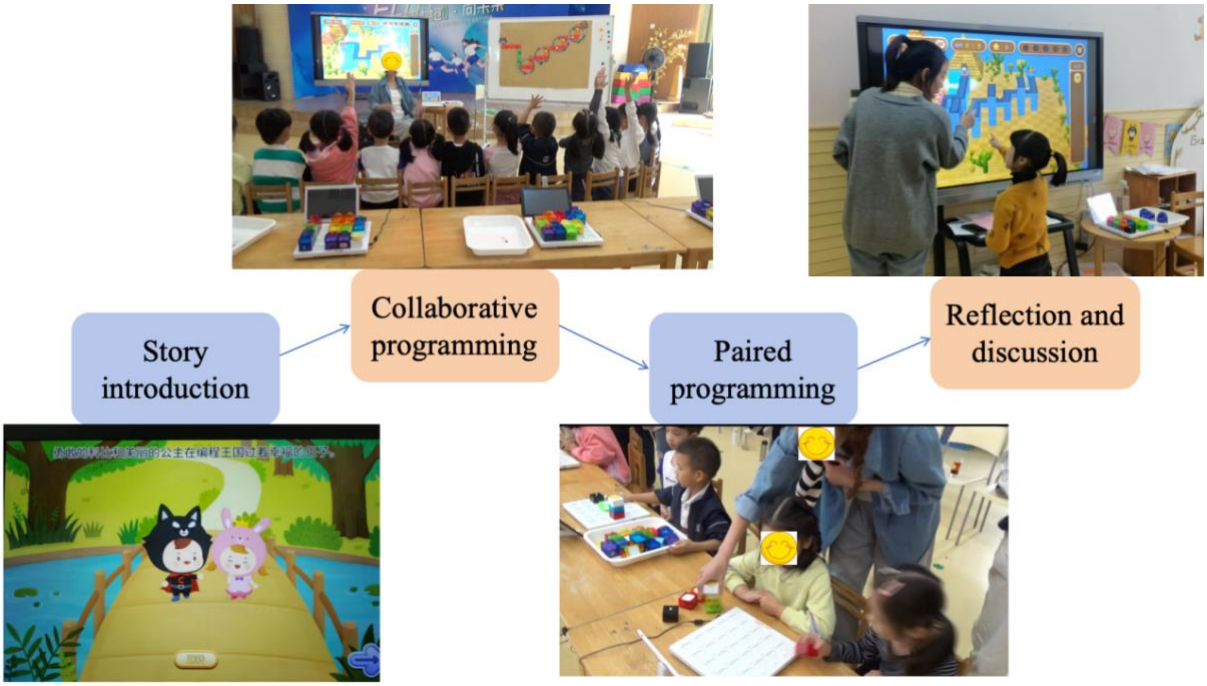
4.2.3.2 Objectives and Content of Programming Activities

The intervention program was designed according to the functions of the programming tool used in this study and the "Early Childhood CT Curriculum Framework"(Zeng et al., 2023a). It encompassed 12 programming activities covering the following content: hardware and software, events, sequences, loops, conditionals, representations, pattern recognition, problem decomposition, debugging, and algorithmic design. The content and objective of each activity, along with corresponding exemplary programming tasks, can be found in the Appendix.

4.2.3.3 Implementation Process of Programming Activities

The programming activities consisted of four segments (see Figure 8):

Figure 8 *Implementation Process of Programming Activities*



(1) Story introduction (2 mins). The teacher told a story to stimulate the children's interest and set the programming context.

(2) Collaborative programming (10 mins). The teacher and children collectively completed a programming task and learned programming concepts and skills in completing the task.

(3) Paired programming with teacher guidance (20 mins). Children worked in pairs, applying programming concepts and skills to complete programming tasks while the teacher observed and guided them.

(4) Reflection and discussion (10 mins). Children reflected and discussed difficulties encountered in programming with the teacher.

4.2.4 Procedure

The study received ethical approval (Reference No. 2021-2022-0315) from the Human Research Ethics Committee of the researchers' University prior to data collection. Informed consent was obtained from the kindergarten principal, the two participating teachers, and the children's parents.

4.2.4.1 The 3-hour Training Session

Before the intervention, the two teachers in the experimental class received a 3-hour training session which aimed to guide them in using the programming tool and conducting the programming activities designed by the researchers.

The training session, lasting three hours, was structured into three sections. The first section aimed to introduce teachers to CT by using relatable examples from children's daily experiences. For example, to elucidate algorithmic thinking, familiar scenarios such as planning a route from home to school, sequencing steps for washing hands, and folding clothes were utilized. These CT components were revisited in the subsequent sections, where teachers interacted with programming tools and explored lesson plans.

The second section of the training course focused on acquainting teachers with the programming tool and guiding them in its utilization. It was designed to be interactive and hands-on, allowing teachers to actively engage with the programming tool. The session began with an explanation of the tool's components and functions. The two teachers were then encouraged to explore the programming tool collaboratively and complete pre-designed programming tasks within it, with demonstrations and guidance provided to help address any challenges they encountered.

The final section introduced the programming course, presenting teachers with twelve pre-designed programming activities, each outlining clear objectives, preparations, and step-

by-step processes. These programming tasks in the 12 activities were aligned with tasks in the MOBLO tool, making the implementation of programming activities easy for teachers. We asked teachers to read the lesson plans and propose their questions and ideas.

4.2.4.2 The Weekly Communication

Furthermore, we kept weekly communication with the two teachers to provide support in addressing any challenges encountered during programming instruction. Each week, I visited the classroom to videotape the programming activities. Following each session, I engaged in discussions with the teachers for approximately 5-10 minutes to review the day's instructional activities. During these conversations, I inquired about the teaching strategies employed by the teacher to understand the rationale behind specific strategies and worked collaboratively to address any challenges they encountered.

4.2.4.3 Challenges the Teachers Encountered

Based on observations and interviews, the teachers have encountered several challenges:

(1) The main challenge for teachers was teaching loops. Children could easily identify simple loops like "forward, right, forward, right," but faced difficulties when encountering a route consisting of a looped path and two segments preceding and following the loop. This complexity made it hard for teachers to effectively support children in recognizing such complex patterns.

(2) It is challenging for teachers to provide tailored scaffolding for children with varying abilities, particularly in delivering specific assistance to help less proficient children challenge complex tasks.

(3) Promoting communication and collaboration among children is another challenge for teachers. In the early stages of the programming activities, teachers noticed that even

though students were paired up, they often programed independently without engaging in meaningful discussions or collaboration with their partner.

(4) The teacher-to-student ratio is relatively high (1:18). Some children seek help from teachers when they encounter problems during programming activities, allowing teachers to identify and address their issues. However, there are also children who struggle silently, making it challenging for teachers to identify and guide them effectively.

4.2.4.4 The Intervention

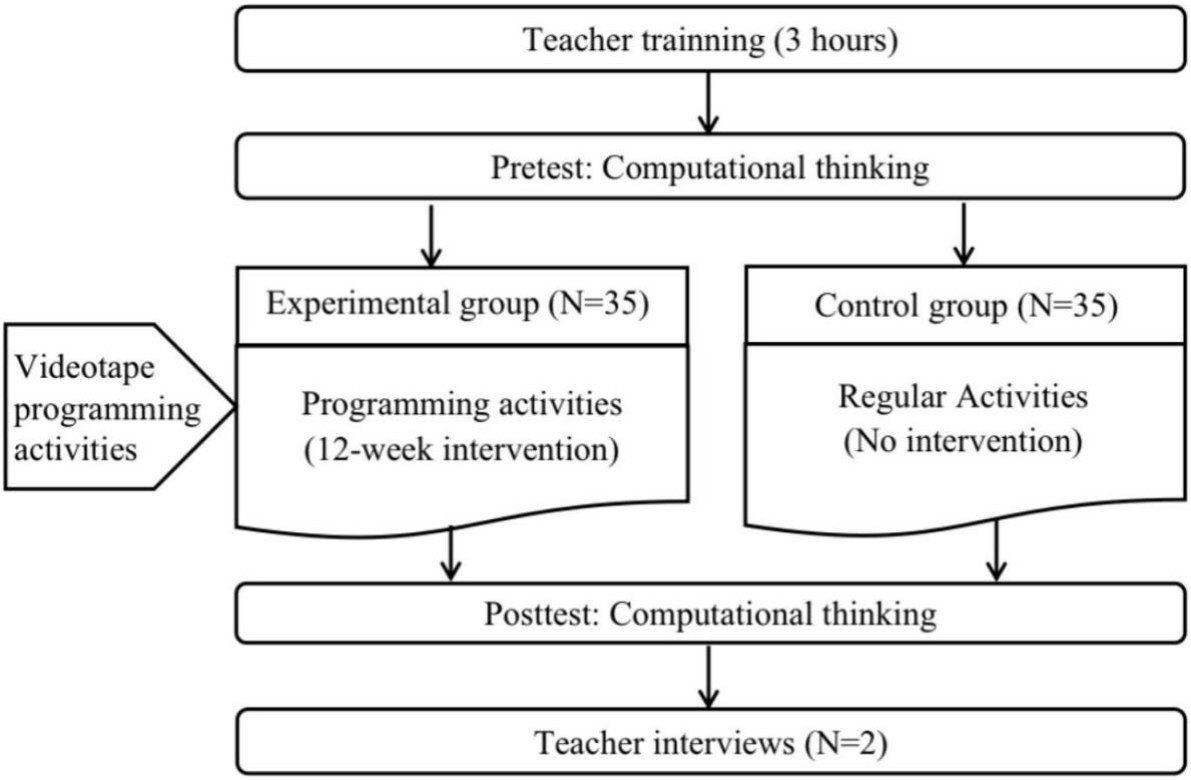
We implemented a 12-week programming intervention in the experimental group (one session per week), with each session lasting approximately 40 minutes. During the same period, the control group children were engaged in learning centre activities without programming elements.

4.2.4.5 Data Collection

All children's CT was measured individually using TechCheck-K in a quiet, open room in the kindergarten before and after the intervention. We recorded the 12 programming activities in one of the groups during the intervention and conducted individual semi-structured interviews with the two teachers from the experimental class upon completion of the intervention.

See Figure 9 for the procedure.

Figure 9 *Procedure*



4.2.5 Data Collection

4.2.5.1 Child Assessment

This study measured children's CT using the TechCheck-K, which consists of 15 multiple-choice items (1 point each) focusing on six CT components: hardware/software control structure, modularity, debugging, representation and algorithms (Relkin et al., 2020). Sample items include “Mice cannot pass through blue walls or red lights. Which mouse will get the cheese?”. In this study, we transformed the TechCheck-K into a digital version of *Wenjuanxing* (a Chinese version of *Qualtrics*) to facilitate its implementation. The pre-test Cronbach's α was 0.812, and the post-test Cronbach's α was 0.803, with good reliability.

4.2.5.2 Videotaped Observations

To investigate children's engagement in programming and teachers' instructional strategies during the programming process, we recorded 12 programming activities, resulting in approximately 900 minutes of video. In the group teaching session, one camera was fixed

in a corner of the classroom to capture the entire teaching process; in the children's programming session, another camera was fixed next to a randomly selected group of two children to record their programming processes. Additionally, the first author hand-held a video camera to track and record the teacher's guidance.

4.2.5.3 Interviews

We conducted individual semi-structured interviews with the two teachers from the experimental class to understand their instructional strategies. The main interview questions were as follows: Do you perceive any challenges in guiding young children to learn programming? What is the primary challenge, if any? What instructional strategies do you employ during programming activities (Why)? Do you find using this strategy effective (How)? The interviews with the two teachers lasted 32 and 35 minutes, respectively, and both were audio-recorded.

4.2.6 Data Analysis

For RQ1, we used SPSS 27.0 software to analyze quantitative data. We first conducted an independent samples t-test to determine if there was a significant difference between the experimental and control classes on the CT pre-test scores. To exclude the potential influence of pretest scores on post-test scores, we then utilized analysis of covariance (ANCOVA) to assess the impact of the intervention on young children's CT.

For RQ2 and RQ3, we first selected videos capturing children's engagement in programming and teachers' instructional strategies (approximately 600 minutes). Subsequently, we transcribed these videos as well as interviews for analysis. Lastly, we analyzed the characteristics of children's engagement in programming and teachers' instructional strategies using the thematic analysis (Braun & Clarke, 2013), which included the following steps:

(1) Becoming familiar with the collected data: I transcribed the video recordings of programming activities related to children's engagement in programming and teachers' instructional strategies.

(2) Generating initial codes: I systematically reviewed the data set to generate initial codes. For example, I generated codes such as "place blocks immediately," "rush to place the Power Blocks," "fail challenges," "careful observation," "pause and think," "record routes using arrows and symbols," "discuss," " simulate the virtual sprite's movements on the screen," "check" … in the data set of children's engagement during the program design stage.

(3) Searching for themes: I looked for commonalities across the codes and combined some codes to form themes. For example, codes such as "place blocks immediately," "rush to place the Power Blocks," and "fail challenges" were grouped under the theme of "programming without cognitive engagement." On the other hand, codes like "careful observation," "pause and think," "record routes using arrows and symbols," "discuss," " simulate the virtual sprite's movements on the screen," and "check" were categorized under the theme of " programming with cognitive engagement."

(4) Reviewing the identified themes: Upon reviewing the initial themes, I found that these two themes accurately reflected the two states observed in children during the programming process.

(5) Defining and labeling the themes: I elaborated "action preceding thought" and "thought before action." "Action preceding thought" refers to the tendency observed in children to manipulate programming blocks without conducting a thorough problem analysis or self-checking of their program beforehand. This behavior leads to issues such as reversing starting and endpoint positions, overlooking known conditions, missing steps, and misidentifying patterns. "Thought before action" is characterized by children's inclination to

engage in problem analysis, pattern recognition, and algorithm design prior to physically manipulating programming blocks. This approach results in the creation of more accurate programs compared to those developed during the initial phase where problem analysis is skipped.

(6) Producing the report: I presented the results in relation to my research questions about children's characteristics of engagement in programming and instructional strategies used by teachers.

4.2.7 Validity of Qualitative Data Analyses

To ensure the validity of the qualitative research findings, member checking and inquiry auditing were employed (Creswell & Creswell, 2017). Member checking involved conducting follow-up interviews with two teachers from the experimental class to ensure consistency with researchers' interpretations and initial responses. Additionally, two researchers specializing in ECE served as auditors to guarantee the rigor of the research and verify the accuracy of the identified video and interview aspects.

4.3 Results

4.3.1 Effect of Programming on Young Children's CT

The results of the independent samples t-test (see Table 8) indicated that there was no significant difference in the CT scores between the experimental and control classes before the intervention (t=-0.09, p>0.05).

Table 8 *A Comparison of CT Pre-Test Scores Between the Experimental Group and Control Group*

	N	Mean	S. D.	*t*	*d*
Experimental group	35	8.26	2.43	-0.09	2.65

Control group	35	8.31	2.86

The results of the ANCOVA (see Table 9) indicated that there was a significant effect of the intervention on young children's posttest CT scores after controlling for pretest CT scores, F (1, 66) = 94.336, p<0.001.

Table 9 *Descriptive Data and ANCOVA of the CT Post-Test Scores*

	N	Mean	S. D.	Std. Error.	Adjusted Mean	F *value*	η^2
Experimental group	35	12.20	1.32	0.10	12.21	94.336***	0.59
Control group	35	8.71	2.93	0.10	8.69		

*** p<0.001

4.3.2 Characteristics of Children's Engagement in Programming

This section analyzed the characteristics of children's engagement in programming, as demonstrated by the two target children, during the program design and debugging stages.

4.3.2.1 Program Design Stage

During the program design stage, children manipulated programming blocks to create instructions. Through an analysis of video recordings of children's programming process in the early learning period (initial four weeks), we identified a key characteristic of their engagement, which we called "action preceding thought." Specifically, children often skipped problem analysis or failed to conduct a detailed problem analysis before manipulating the programming blocks. Additionally, they often overlooked self-checking their program before executing it. These tendencies resulted in various problems, such as reversing the starting point and endpoint positions, overlooking known conditions, missing steps, and

misidentifying patterns. The following observation transcription reflects the characteristic of "action preceding thought" in the targeted children.

> *In the Stage 3 Level 4 task of the "Sequence" software (referring to the programming task example in Activity 3 in the Appendix), the objective is to guide a virtual sprite along a designated route and reach the destination. However, the route has two broken sections. To overcome these obstacles, the children must first obtain the 'wings' prop and then use it to fly over the obstacles. However, in their initial attempts, Child A and Child B neglected to analyze the number of broken sections on the path or consider the required number of 'wings' props. Consequently, they made incorrect decisions by directly moving forward in the direction the virtual sprite was facing, thus missing the first 'wings' prop. Additionally, after arranging the programming blocks, both Child A and Child B executed the program without conducting a prior check.*

The video analysis revealed a notable shift in the engagement of the target children during the later stages of their learning period (the last four weeks) compared to the early stages. We labeled this characteristic as "thought before action." In this stage, they tended to analyze and decompose problems, recognize patterns, and design algorithms before physically manipulating the programming blocks. Consequently, the programs created during this stage were more accurate compared to those created during the early intervention. The following transcript of observations illustrates this "thought preceding action" characteristic in the target children.

> *In Stage 3, Level 4 of the "Loops" software (referring to the programming task example in Activity 9 in the Appendix), the virtual sprite follows a route consisting of a looped path and two segments preceding and following the loop. Instead of*

immediately reaching for the programming blocks, the two children took a more thoughtful approach. They used their index fingers to simulate the virtual sprite's movements on the screen while verbally describing them, such as "to the right, up... " (demonstrating attentive observation and problem analysis). Subsequently, they took out paper and pens, with Child A verbally describing the virtual sprite's path as Child B recorded it using arrows and symbols (showcasing their collaborative efforts). Then, they audibly read out the recorded arrows and symbols. Consequently, they recognized the loop unit, marked it with a line underneath, and noted the number of loops (showcasing their ability to recognize the pattern utilizing the paper provided by their teacher and employing "rhythmic reading" techniques). Afterward, Child A and Child B took out the programming blocks and began coding. They used Direction Blocks, Action Blocks, NFC Blocks, and Number Cards to create instructions for the looped path. Using stacking, they arranged the blocks for the segments before and after the loop. Then, they positioned the three stacks, representing the three segments of the route (looped path, segment before it, and segment after it) in order on the sensor board (demonstrating their problem decomposition skills). Upon arranging the programming blocks, instead of rushing to place the Power Blocks, Children A and B simulated the virtual sprite's actions step by step using their extended index fingers according to their devised program. After confirming the accuracy of their program, they placed the Power Blocks on the Sensor Board (check their designed program before starting it).

4.3.2.2 Program Debugging Stage

In the program debugging stage, the virtual sprite executes the received instructions, and if the instructions fail to execute successfully, the children need to debug the program

continuously (by removing redundant instructions, adding missing instructions, adjusting out-of-order instructions, etc.) and re-validate it until it succeeds. Video analysis revealed that during the early learning period (the initial four weeks), the targeted children demonstrated a tendency that we labeled as "relying on trial-and-error". Specifically, when the virtual sprite executed the program, the children merely "watched" the sprite's actions without intending to correlate them with the program they had created; after the virtual sprite stopped moving, the children lacked awareness of analyzing the errors based on the sprite's stopping position and modifying the program accordingly. Instead, they attempted to complete the task through repetitive trial-and-error. The following transcription reflects this characteristic in the targeted children.

> *When Child A and Child B saw that the virtual sprite did not reach the endpoint, Child B exclaimed, "We failed. Let's start again!" So, they immediately removed all the blocks from the Sensor Board and made another attempt. After three unsuccessful tries, Child A sought assistance from the teacher.*

However, in the later stages (the last four weeks), the targeted children demonstrated the ability of "active debugging." Specifically, they exhibited attentive observation of the virtual sprite's actions on the screen and compared them consciously with their own created programs. When the virtual sprite ceased its movements, the children attempted to identify errors in the program based on the position where it stopped and then modified the program to verify it until it succeeded. The children experienced an iterative cycle of "observation, recognition, modification, and validation" throughout the debugging process. The following transcription reflects the characteristic of young children's "active debugging."

> *In Stage 3, Level 4 of the "Loops" software (refer to the programming task example in Activity 9 in the Appendix), although Child A and Child B had carefully analyzed the*

problem before arranging the programming blocks, the program they created still had some minor issues. Upon observing the virtual sprite stopping after completing the looped path, both children expressed a perplexed "hmm..." sound and displayed expressions of confusion. Subsequently, they fixed their gaze on the screen, looking at the spot where the virtual sprite had stopped. Child A pointed to where the virtual sprite had stopped and looked at the programmed blocks they had created. Suddenly, Child A exclaimed, "Next, it should go upwards, upwards!" Child B seemed to notice something and excitedly shouted, "Yes! Yes!" Child A then rushed off to find the Upward Direction Block and inserted it in front of the rightward and downward direction blocks. Child B took out the Power Block and put it at the end. Both children firmly held the Power Block, their eyes fixed on the screen, carefully watching the virtual sprite move step by step according to the program. When the virtual sprite reached the endpoint, defeating the Monster, both children joyfully exclaimed, "Ah! We did it! "

4.3.3 Teachers' Instructional Strategies in Programming Activities

According to the video and interview data analysis, teachers mainly utilized three instructional strategies to support children's programming. These strategies were named "guiding children to observe closely," "guiding children to pause," and "providing children with external scaffolding for thinking."

4.3.3.1 Guiding Children to Observe Closely

During the program design stage, teachers guided children to observe closely. They used demonstrations and verbal prompts to encourage careful problem analysis before manipulating programming blocks. This involved observing starting and ending positions, identifying patterns in the route, recognizing obstacles and existing clues on the path and so

on. The following observation transcript illustrates the teacher's use of this method.

(Group Session) Teacher: *Today, as usual, we will first plan the route on the blackboard and then arrange the programming blocks on the Sensor Board. Now, please observe the positions of Kobe and the princess. Kobe is here* (pointing to Kobe), *and the princess is here (pointing to the princess).*

Teacher: *This route is a bit different from what you usually see. What does Kobe need to pass through to rescue the princess?*

Child: *A tunnel.*

Teacher: *What tool does he need to pass through the tunnel?*

Child: *A mushroom.*

Teacher: *So, we need to get the mushroom before passing through the tunnel. Can we pass if we don't get the mushroom and try to go through the tunnel directly?*

Child: *No, we can't.*

Teacher: *So, what should be Kobe's first step?*

Child: *Go up.*

Teacher: *That's right* (The teacher draws "↑" on the blackboard). *What comes next?*

Child: *Go left* (The teacher draws "←" on the blackboard).

Child: *Then go right.*

Teacher: *Why do we need to go right?*

Child: *Because after getting the mushroom, we must come back out.*

Teacher: *You've thought it through carefully* (The teacher draws "→" on the blackboard)

... (The teacher guides the children to observe and verbalize each of Kobe's movements, and then the teacher records the route on the blackboard using arrows. The final

recorded route is: ↑ ← → ↑ → ↓C ↓↓C↓↓C↓) (C represents the Action Block).

Teacher: *Just now, as you said, I recorded, and we have written down the entire route Kobe takes to rescue the princess. Now, please look for patterns in this route. Carefully observe, which part of this route is repeated.*

(The teacher guides the children to identify the distinctive C, then observe the parts before and after C, and finally identify the loop unit "↓C↓". The teacher circles this part, asks the children to count how many times it repeats, and records the number.)

Teacher: *Now, I'll have one of you try to arrange the programming blocks following the route we recorded on the blackboard...*

In the program debugging stage, teachers guided children to closely observe the virtual sprite's actions and the illuminated programming blocks that corresponded to the sprite's actions. They used demonstrations and verbal prompts to establish a connection between the two. When the virtual sprite stopped, teachers guided the children to identify errors based on its stopping position and modify the program accordingly.

4.3.3.2 Guiding Children to Pause

During the program design stage, teachers used demonstrations, verbal prompts, and other methods to encourage children to pause and simulate the virtual sprite's actions based on their created program. This approach allowed children to review and check the programs they had created before applying the Power Blocks to run the program. Teachers pointed out in the interview that children were often eager to see the virtual sprite execute the program, making them less careful and patient in creating the program. By "pausing the placement of Power Blocks" and "simulating the virtual sprite's actions", children can better understand the correspondence between actions and instructions and check program correctness.

During the program debugging stage, guiding children to pause involved teachers

advising them not to immediately remove blocks or modify the program when the virtual sprite ceased its actions on the screen. Instead, teachers encouraged children to identify errors in the program based on the feedback provided by the virtual sprite. According to teachers' interviews, children initially tended to remove all the blocks and start programming again in response to program failure, leading to repetitive mistakes. The practice of pausing the removal of blocks proved beneficial as it helped children identify program errors and enhanced debugging efficiency.

4.3.3.3 Providing External Scaffolding for Thinking

Providing external scaffolding for thinking was an important practice the teachers used to support children in learning loops. Firstly, the teachers provided children with paper and pens to draw the virtual sprite's path step by step. Afterward, the children examined the drawn path, analyzed the loop units, and determined the number of loops in the path. Then, they represented these loops by circling, drawing lines, and writing numbers on the paper. Finally, they used programming blocks to create programs based on the notes they had made on the paper. The utilization of this strategy is also demonstrated in the "Guiding Children to Observe Closely" section.

Teachers explained in interviews that this strategy was employed due to children's challenges in accurately identifying loop units within a route. By recording the path on paper, children could clearly see each step and identify loop units by comparing, circling, and drawing lines. Additionally, video and interview analyses revealed that this strategy fostered the development of careful observation and problem analysis before manipulating programming blocks. Furthermore, it facilitated communication and discussion among children as they engaged in conversations about "whether the paths recorded on the paper are correct," "how to find the loop units," and " whether they have placed the programming

blocks according to the paper records."

4.4 Discussion

The study revealed a significant improvement in young children's CT after participating in a 12-week programming program. This aligns with previous research that also demonstrated the effectiveness of programming education in fostering CT among young children (Angeli & Valanides, 2020; Bers et al., 2019; Relkin et al., 2021; Yang et al., 2023). However, prior studies primarily utilized physical, virtual, and hybrid kits with virtual programming blocks as programming tools (Yu & Roque, 2019). This study is the first to utilize a hybrid kit with tangible programming blocks for teaching programming to young children, confirming its positive impact on enhancing children's CT. The values and virtues of this hybrid programming kit lie in various forms of feedback it offers, both from the physical programming environment and the game interface. In the tangible programming environment, the virtual sprite executes the program, and the corresponding electronic programming blocks light up at each step, allowing children to visually connect the sprite's actions with their own program. The game interface provides feedback through the virtual sprite's execution of the program. This helps children to identify errors in their program and practice children's debugging skills, an important aspect of CT (Zeng et al., 2023a). Additionally, the programming instruction record bar displays instructions entered by the child in real-time, which helps the child associate the tangible programming blocks with the symbolic representations, thus facilitating the learning of "representation". Furthermore, teachers do not have to design programming tasks, since different difficulty levels of programming tasks have already been embedded in this hybrid programming kit to develop children's various CT skills. After successfully completing a task, the children can seamlessly transition to the next level.

Moreover, this research focused on the process of young children's programming, providing a new perspective for understanding programming as a means to promote CT. We found that, initially, at the programming design stage, children's engagement in programming was characterized by "action preceding thought", i.e., manipulating programming blocks without careful problem analysis. Qu and Fok (2021) also identified a similar trend among some students (aged 7-9) who tended to operate programming blocks without critically analyzing problems or making thoughtful decisions. Notably, these students demonstrated less improvement in their CT scores compared to those who engaged in careful analysis and exercised caution in their decision-making. Additionally, during the program debugging stage, we observed that children initially relied on a "trial-and-error" approach. Chevalier et al. (2022) found similar results in educational robotics learning activities, where children dependent on immediate feedback without guidance mainly used trial-and-error strategies, reducing their cognitive engagement.

However, this study revealed that under the guidance of teachers, children's engagement in programming transitioned from "action preceding thought" to "thought preceding action" and from "relying on trial-and-error" to "active debugging." This finding not only validates the positive impact of programming education on children’s CT but also highlights the essential role of teachers' guidance in children's programming and CT learning. Wang et al. (2020) also emphasized the crucial importance of teachers' scaffolding in enabling young children to easily understand and engage with CT. Without teachers' guidance, students may lose interest in programming activities and struggle to demonstrate an understanding of algorithmic design (Newhouse et al., 2017). Additionally, students may rely on trial-and-error rather than actively engage in reflective problem-solving (Biesta & Burbules, 2003).

We found that the teachers mainly employed three strategies, namely "guiding children to observe closely," "guiding children to pause," and "providing children with external scaffolding for thinking" to enhance children's programming and learning. This finding aligns with the observations made by Wang et al. (2020), who also identified effective strategies employed by a teacher to foster children's CT. These strategies included "modeling a systematic way of checking and identifying a problem" and "encouraging children to pause and assess" (Wang et al., 2020, p. 14), which correspond to the "guiding children to observe closely" and "guiding children to pause" strategies implemented by the teachers in our study. Furthermore, Angeli and Valanides (2020) demonstrated the efficacy of the "external memory scaffolding" strategy in cultivating young children's CT. Likewise, other researchers emphasized the importance of providing scaffolding to assist young children in mastering challenging commands and longer, more intricate sequences, thereby overcoming their memory constraints (Macrides et al., 2021). These findings are consistent with the "external scaffolding for thinking" strategy employed by the teachers in this study.

4.4.1 Limitations and Future Research

This study exhibits certain limitations that indicate future research directions. Firstly, the small sample size of children being observed limits the generalizability of the findings. This study involved a 12-week longitudinal observation of a randomly selected group of two children, aimed at investigating children's engagement in programming. While the two children's programming processes were analyzed rigorously, objectively reflecting the characteristics of their engagement in programming, prudence is advised when extrapolating the findings to a broader population of children. To address this issue, future research should include larger and more diverse samples of young children.

Moreover, it is essential to note that the findings on the instructional strategies were

based on observations of one teacher and interviews with two teachers from the experimental class. Although we engaged in weekly dialogues with the two teachers to understand their challenges and strategies in programming teaching, and our analysis of observations and interviews was comprehensive and rigorous, caution is advised when generalizing these research findings to other early childhood educators. Future studies should encompass a broader representation of teachers with varying levels of proficiency in early childhood pedagogy. Such studies can shed light on the professional development needs of early childhood teachers aiming to introduce CT education in their classrooms (Wang et al., 2020).

4.4.2 Contributions and Implications

This study makes both methodological and practical contributions. Methodologically, the study adopted a mixed-methods approach, combining pre-and post-tests to assess changes in children's CT scores, longitudinal observations to understand the characteristics of children's engagement in programming, and interviews and observations to gain insights into teachers' support for children's programming. The mixed-methods approach addresses a common limitation in quasi-experimental research, which often focuses solely on children's learning outcomes without considering their learning process or the support provided by teachers.

Practically, this study confirmed the positive impact of using hybrid kits with virtual sprites and tangible programming blocks in promoting CT among young children, offering insights for educators in selecting appropriate programming tools. Furthermore, it revealed the characteristics of children's engagement during program design and debugging stages, providing valuable insights for teachers to offer targeted support. Finally, this study emphasized the crucial role of teachers in supporting young children's programming and demonstrated that, through proper training, teachers could adopt targeted strategies to

effectively enhance young children's CT learning. This implies that educational institutions and policymakers should offer programming and CT education training programs to integrate programming education in early childhood settings and cultivate future generations' essential CT skills for the digital age.

Chapter 5: General Discussion and Conclusions

5.1 Limitations and Future Research Directions

The overarching objective of this research is to investigate the fundamental aspects of "what to teach" and "how to teach" in the domain of early programming and CT, which hold significance across all disciplines. The first study has developed a CT curriculum framework for ECE that addresses the query of "what to teach". The second study explored an early childhood teacher's CK and PK in early programming and CT. The identification of teachers' areas of weak knowledge, misconceptions, and teaching challenges provides insights into the question of "how to teach". Additionally, the third study focuses on another critical aspect of programming and CT education, which is the evaluation of programming tools. This study also contributes valuable insights into the question of "how to teach".

However, programming and CT education in ECE is still in its early stages, and there are numerous theoretical and practical issues that require further exploration. This section outlines a list of possible future works, focusing on what to teach, how to teach, whom to teach, how to evaluate, teacher professional development, CT's role in early learning and development, and family engagement.

5.1.1 What to Teach

Understanding CT learning trajectories for different developmental stages is critical to improving the effectiveness of CT education. However, the CT curriculum framework for ECE fails to specify which concepts, practices and perspectives children of different ages (between 2 and 8 years old) should learn and what developmental level they can achieve. Therefore, future research should focus on studying CT learning trajectories for young children to help practitioners understand the learning and developmental characteristics of young children's CT in order to develop age-appropriate learning goals.

5.1.2 How to Teach

The methods employed in teaching programming and CT significantly impact young learners. Future studies should compare the effectiveness of various pedagogical approaches, such as project-based learning, game-based learning, and direct instruction. Additionally, instructional strategies like unplugged activity, embodied cognition, external memory support scaffolding, pair programming should be examined to determine their influence on children's CT development. Equally important is evaluating the impact of different programming tools, such as plugged versus unplugged and physical kits versus virtual kits, to identify which tools best support young learners' understanding and engagement.

5.1.3 Whom to Teach

Determining the optimal age for introducing programming and CT education is another critical research direction. Studies should investigate the developmental readiness of children at various ages, identifying the age at which they can most effectively begin to learn these skills. Furthermore, understanding the developmental trajectory and characteristics of children's CT learning can help tailor educational approaches to their cognitive abilities. Research should also focus on identifying common misconceptions children have about programming and CT, allowing educators to address these issues proactively.

5.1.4 How to Evaluate

Effective evaluation methods are crucial for comprehending and promoting children's programming and CT skills. However, the commonly used tool in current research for assessing young children's CT is the TechCheck-K. This tool evaluates CT in children aged 5-9 through tasks like problem-solving, sequencing, graph decomposition, pattern recognition, determining the shortest path, and navigating an obstacle course maze. While this tool addresses the issue of young children needing a programming foundation for CT assessment,

thus offering a practical approach to assessing CT in young children (Relkin & Bers, 2021), it is merely an outcome-based evaluation and does not provide a formative assessment tool to observe the development of children's CT and programming abilities in real-life contexts. Future research should focus on developing assessment tools that can accurately gauge students' CT skills. These tools might include formative assessments, performance-based tasks, and observational protocols that provide insights into children's thought processes and problem-solving strategies.

5.1.5 Teacher Professional Development in Early Programming and CT Education

The research has shown that teachers' support plays a crucial role in young children's programming and CT learning. At the same time, this research have indicated a severe lack of PCK and confidence in the field of programming and CT education among teachers (Zeng et al., 2024). Therefore, acquiring the relevant PCK and pedagogical skills is gradually emerging as a new demand placed on educators (Yadav et al., 2016) and providing professional development support for teachers to help them successfully integrate CT education into their curriculum has become an urgent need.

An increasing number of professional development programs aim to provide teachers with training in programming and CT instruction, but they often target more primary and secondary education teachers rather than those in ECE. An example of professional development with a focus on CT is the CT4EDU project in the United States. This initiative, supported by the National Science Foundation, involves collaboration between Michigan State University, Oakland Schools, and the American Institute for Research. The goal is to create and implement a high-quality curriculum and professional development program to assist elementary school teachers in integrating CT into their classrooms. In addition to creating teaching resources like posters and lesson screeners, the project has also conducted

research on effective professional development for CT (Rich, Yadav, and Larimore, 2020; Rich, Yadav, and Schwarz, 2019). Future research could focus on developing training programs for early childhood teachers in programming and CT education, and conducting empirical research on the effectiveness of these programs.

5.2 Implications

The three studies contribute to the theoretical understanding of CT education, provide practical insights for teachers and teacher educators, and offer recommendations for policy development. These contributions collectively enhance the field of CT education in early childhood.

5.2.1 For policymakers

Clarifying the importance and feasibility of early programming and CT education can assist policymakers in promoting policies to advance programming and CT education in early childhood education.

The importance of CT as an essential skill for the 21st century is widely acknowledged. However, many regions and countries do not currently include CT in their ECE policies. Offering initial guidance on what should be taught and how it should be taught can assist policymakers in formulating curriculum guidelines that specify objectives, content, implementation strategies, and other relevant aspects. This, in turn, can facilitate the advancement and dissemination of CT education in early childhood environments. This will also provide practitioners with a strong foundation for their professional endeavors.

Additionally, the research highlights the importance of programming tools in helping young children learn CT. This can guide policymakers in providing the necessary funding and resources to support practitioners in implementing programming and CT education in kindergarten.

Finally, this research underscore the importance of educational institutions and policymakers providing training programs in programming and CT education for early childhood teachers. This highlights the necessity for policy support and investment in professional development to enable teachers to effectively integrate programming and CT education in early childhood settings.

5.2.2 For early childhood practitioners (leaders and teachers)

The research defined the content of programming and CT education, helping practitioners better understand CT education and the importance of implementing it in kindergarten. It provides a curriculum framework for programming and CT education, analyzes effective and ineffective teaching strategies, and outlines characteristics of children's programming learning. All these practical guidelines assist practitioners in implementing programming and CT education effectively.

5.2.3 For teacher educators and teacher education institutions

The research highlights the crucial role of teachers in supporting young children's programming and CT learning and reveals the insufficient knowledge of teachers in programming and CT education. Therefore, it provides strong evidence for offering professional support to educators. Furthermore, the study clearly identifies specific deficiencies in teachers' CK and PK, providing a solid foundation for teacher educators to offer targeted and effective professional support.

5.2.4 For future research

The CT framework established in this study can serve as a basis for analyzing teachers' content knowledge and designing intervention programs in the future. Additionally, methods employed in this study can provide insights for future research. For example, the second study using the CK and PK framework to analyze individual teachers' CK and PK can

be a reference for future research to expand the sample size and further explore teachers' CK and PK comprehensively and in-depth; the third study adopted a mixed-methods approach, addressing a common limitation in quasi-experimental research, which often focuses solely on children's learning outcomes without considering their learning process or the support provided by teachers.

In conclusion, this study has provided initial answers to how to conduct programming and CT education effectively in the early years, facilitating the advancement of programming and CT education in early childhood education as well as research in this field.

5.3 Extension of Research in ECE and Computing Education

5.3.1 Extension of Research in ECE

This research introduces programming and CT education into the realm of ECE, significantly extending the line of research in the field of ECE. Firstly, study 1 established a CT curriculum framework for ECE. This framework provides a solid foundation for developing programming and CT education and research in ECE because it answers the question of “what to teach” in early programming and CT and facilitates future studies to be conducted within a unified CT curriculum framework for ECE.

Secondly, Study 2 expands the existing research on PCK of early childhood teachers to the domain of programming and CT. While previous studies have examined early childhood teachers' PCK in areas such as language, mathematics, science, health, and arts, there has been a dearth of research on PCK in the field of programming and CT. The construction of CK and PK frameworks for early programming and CT education, established through a thorough analysis of existing literature, offers valuable frameworks for future research on early childhood teachers' PCK in programming and CT. Furthermore, the exploration of early childhood teachers' PCK in early programming and CT provides valuable

insights for future research on investigating effective teacher training programs for early childhood teachers in early programming and CT. Building upon the weak knowledge areas, misconceptions, and teaching difficulties identified among teachers, we can design a training program to promote early childhood teachers' PCK of programming and CT and verify the effectiveness of the training program through quasi-experimental studies.

Thirdly, study 3 focused on another crucial factor in programming and CT education: programming tools. Specifically, the study investigated the impact of a hybrid programming tool on children's CT skills and analyzed the programming behaviors displayed by children when using this tool. Future research could compare the effectiveness of different types of tools, such as comparing the effects of plugged and unplugged programming tools, on young children's CT. This comparative analysis would provide empirical evidence to guide the selection of suitable programming tools for young children.

In addition, this thesis provides an initial investigation of the learning characteristics exhibited by young children during the program design and debugging stages. While previous studies have explored the learning characteristics of young children in domains such as language, mathematics, science, and arts, there has been a lack of research on the learning characteristics of young children in programming and CT. Moreover, previous research has primarily focused on the learning outcomes of children's programming education, allocating less attention to the learning process itself. This research extends the line of research by exploring children's programming learning processes and learning characteristics. Future research can further delve into the learning trajectory, learning characteristics, and the levels of learning and development that can be achieved by young children at different age ranges in programming and CT. This will provide a foundation for teachers to adopt targeted instructional strategies.

Lastly, this thesis offers an initial exploration of instructional strategies to facilitate young children's learning of programming and CT. Similarly, research in this area remains limited. Future studies can delve further into teachers' instructional strategies to support young children's learning of programming and CT.

5.3.2 Extension of Research in Computing Education

This thesis further extends the line of research in the field of computing education. Firstly, it broadens the scope of computing education research to encompass the realm of ECE. While considerable research has been conducted on computing education in primary and secondary education, the research on ECE remains relatively limited. Specifically, there is a lack of consensus on which components of CT should be taught in ECE. Study 1 addresses this gap by developing a refined CT curriculum framework for ECE, which provides a solid foundation for future research in early programming and CT education.

Moreover, Study 2 expands the line of research in computing education by examining teachers' PCK in early programming and CT. The investigation of teachers' PCK holds significant implications for enhancing their professional knowledge and improving their teaching effectiveness, thereby advancing the field of computing education.

Furthermore, Study 3 expands the existing body of research in the field of computing education by investigating the programming behaviors exhibited by children during the stages of program design and debugging. Previous studies have predominantly focused on examining the learning outcomes of children's programming education, with limited attention being paid to the actual learning process itself. By specifically examining the process of computing education, it is possible to gain a deeper understanding of the trajectory and characteristics of computer science learning, as well as the levels of learning and development that students at different age groups can achieve. Consequently, this knowledge

can be utilized to provide more targeted support for children, thereby facilitating the advancement of computing education.

5.4 Overall Framework for Early Childhood CT Education

Based on the aforementioned information, I have summarized the following framework for early childhood programming and CT education:

1. What to teach: The content to be taught in early CT education, as delineated in The Early Childhood CT Framework (see Table 3), includes the CT concepts, practices, and perspectives that are appropriate for young children to learn.

2. How to teach: The instructional methods encompass the teaching context, pedagogical approaches, activity structure, pedagogical strategies, and programming tools employed to foster children's programming and CT skills, as presented in The Programming and CT Pedagogical Knowledge Framework in ECE (see Table 5).

3. Whom to teach: This comprises several considerations: a) determining the most suitable age to introduce programming and CT education, b) understanding the developmental trajectory and unique characteristics of young children's programming and CT skills, and c) identifying prevalent misconceptions in children's learning of programming and CT. However, these specific areas have not been exhaustively examined in the current thesis. Future research endeavors should focus on investigating these aspects to advance our comprehension of young children's programming and CT learning, thereby enhancing the efficacy of CT instruction for young children.

4. How to assess: Assessment is a crucial component of teaching, along with instructional content, methods, and learners. In this thesis, the TechCheck-K tool was employed to evaluate young children's CT. This tool utilizes an unplugged approach, effectively addressing the requirement of prior programming experience in previous CT

assessments. However, while this tool is suitable for research purposes, its applicability in early childhood CT education may be limited. Assessing the CT of the same child at different developmental stages using this tool may result in repetitive measurement effects, and the measurement results may also be influenced by the ongoing development of children's thinking. Employing "formative assessment" to evaluate children may be the most suitable approach, as it relies on observation and communication during children's daily activities. Through careful observation, teachers can discern children's performance and developmental levels in various CT components, thus enabling targeted instruction that aligns with their abilities. Consequently, the development an observational scale for teacher to assess CT in young children would be advantageous, and it holds promise as a future direction for research efforts.

References

Ahn, J., Sung, W., & Black, J. B. (2021). Unplugged debugging activities for developing young learners' debugging skills. *Journal of Research in Childhood Education*, 1-17. https://doi.org/10.1080/02568543.2021.1981503

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, *19*, 30-55. https://doi.org/10.1016/j.ijcci.2018.10.004

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, *105*, 105954. https://doi.org/10.1016/j.chb.2019.03.018

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society*, *19*(3), 47-57. https://www.jstor.org/stable/pdf/jeductechsoci.19.3.47.pdf

Bakala, E., Gerosa, A., Hourcade, J. P., & Tejera, G. (2021). Preschool children, robots, and computational thinking: A systematic review. *International Journal of Child-Computer Interaction*, *29*, 100337. https://doi.org/10.1016/j.ijcci.2021.100337

Ball, D. L., & McDiarmid, G. W. (1989). *The subject matter preparation of teachers*. National Center for Research on Teacher Education East Lansing.

Bati, K. (2021). A systematic literature review regarding computational thinking and programming in early childhood education. *Education and Information Technologies*. https://doi.org/10.1007/s10639-021-10700-2

Bers, M. U. (2018). *Coding as a playground: Programming and computational thinking in*

the early childhood classroom. Routledge.

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

Bers, M. U., González-González, C., & Armas–Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education*, *138*, 130-145. https://doi.org/10.1016/j.compedu.2019.04.013

Bers, M. U., Strawhacker, A., & Sullivan, A. (2022). The state of the field of computational thinking in early childhood education. https://doi.org/10.1787/19939019

Biesta, G., & Burbules, N. C. (2003). *Pragmatism and educational research*. Rowman & Littlefield Publishers.

Block, R. A., Hancock, P. A., & Zakay, D. (2010). How cognitive load affects duration judgments: A meta-analytic review. *Acta psychologica*, *134*(3), 330-343. https://doi.org/10.1016/j.actpsy.2010.03.006

Bower, M., & Falkner, K. (2015). Computational thinking, the notional machine, pre-service teachers, and research opportunities. Proceedings of the 17th Australasian Computing Education Conference, Sydney, Australia.

Braun, V., & Clarke, V. (2013). *Successful qualitative research: A practical guide for beginners*. Sage publications.

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, Vancouver, Canada.

Çakıroğlu, Ü., & Kiliç, S. (2020). Assessing teachers' PCK to teach computational thinking via robotic programming. *Interactive Learning Environments*, 1-18. https://doi.org/10.1080/10494820.2020.1811734

Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, *17*, 93-100. https://doi.org/10.1016/j.ijcci.2018.06.005

Chan, S. W., Looi, C. K., Ho, W. K., & Kim, M. S. (2022). Tools and Approaches for Integrating Computational Thinking and Mathematics: A Scoping Review of Current Empirical Studies. *Journal of Educational Computing Research*, Article 07356331221098793. https://doi.org/10.1177/07356331221098793

Chevalier, M., Giang, C., El-Hamamsy, L., Bonnet, E., Papaspyros, V., Pellet, J.-P., Audrin, C., Romero, M., Baumberger, B., & Mondada, F. (2022). The role of feedback and guidance as intervention methods to foster computational thinking in educational robotics learning activities for primary school. *Computers & Education*, *180*, 104431. https://doi.org/10.1016/j.compedu.2022.104431

Cho, Y., & Lee, Y. (2017). Possibilities of improving computational thinking through activity based learning strategy for young children. *Journal of Theoretical & Applied Information Technology*, *95*(18). http://www.jatit.org/volumes/Vol95No18/6Vol95No18.pdf

Çiftci, S., & Bildiren, A. (2020). The effect of coding courses on the cognitive abilities and problem-solving skills of preschool children. *Computer Science Education*, *30*(1), 3-21. https://doi.org/10.1080/08993408.2019.1696169

Clarke-Midura, J., Silvis, D., Shumway, J. F., Lee, V. R., & Kozlowski, J. S. (2021). Developing a kindergarten computational thinking assessment using evidence-centered design: the case of algorithmic thinking. *Computer Science Education*, *31*(2), 117-140. https://doi.org/10.1080/08993408.2021.1877988

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), 37-46. https://doi.org/10.1177/001316446002000104

Creswell, J. (2014). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. 4th ed*. Thousand Oaks, CA: SAGE.

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

Critten, V., Hagon, H., & Messer, D. (2022). Can pre-school children learn programming and coding through guided play activities? A case study in computational thinking. *Early Childhood Education Journal*, *50*(6), 969-981. https://doi.org/10.1007/s10643-021-01236-8

CSTA, & ISTE. (2011). *Operational Definition of Computational Thinking for K-12 Education.*

Cutumisu, M., Adams, C., & Lu, C. (2019). A Scoping Review of Empirical Research on Recent Computational Thinking Assessments. *Journal of Science Education and Technology*, *28*(6), 651-676. https://doi.org/10.1007/s10956-019-09799-3

Dasgupta, A., Rynearson, A. M., Purzer, S., Ehsan, H., & Cardella, M. E. (2017). Computational Thinking in Kindergarten: Evidence from Student Artifacts (Fundamental)[J]. American Society for Engineering Education, Columbus, OH.

Del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational

thinking through unplugged activities in early years of Primary Education. *Computers & Education*, *150*, 103832. https://doi.org/10.1016/j.compedu.2020.103832

Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education*, *46*(3), 277-296. https://doi.org/10.1080/15391523.2014.888272

Di Lieto, M. C., Inguaggiato, E., Castro, E., Cecchi, F., Cioni, G., Dell'Omo, M., Laschi, C., Pecini, C., Santerini, G., & Sgandurra, G. (2017). Educational Robotics intervention on Executive Functions in preschool children: A pilot study. *Computers in Human Behavior*, *71*, 16-23. https://doi.org/10.1016/j.chb.2017.01.018

Dietz, G., Landay, J. A., & Gweon, H. (2019). Building blocks of computational thinking: Young children's developing capacities for problem decomposition. CogSci, Montreal, Canada.

Dunekacke, S., & Barenthien, J. (2021). Research in early childhood teacher domain-specific professional knowledge - a systematic review. *European Early Childhood Education Research Journal*, *29*(4), 633-648. https://doi.org/10.1080/1350293x.2021.1941166

Ehsan, H., Rehmat, A. P., & Cardella, M. E. (2021). Computational thinking embedded in engineering design: capturing computational thinking of children in an informal engineering design activity. *International Journal of Technology and Design Education*, *31*(3), 441-464. https://doi.org/10.1007/s10798-020-09562-5

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education. Innovations in Practice*, *13*, 153-169. http://www.jite.org/documents/Vol13/JITEv13IIPvp153-169Elkin882.pdf

Elkin, M., Sullivan, A., & Bers, M. U. (2016). Programming with the KIBO Robotics Kit in Preschool Classrooms. *Computers in the Schools*, *33*(3), 169-186. https://doi.org/10.1080/07380569.2016.1216251

Ezeamuzie, N. O., & Leung, J. S. C. (2022). Computational Thinking Through an Empirical Lens: A Systematic Review of Literature. *Journal of Educational Computing Research*, *60*(2), 481-511, Article 07356331211033158. https://doi.org/10.1177/07356331211033158

Fadjo, C. L. (2012a). *Developing computational thinking through grounded embodied cognition* [Columbia University]. ProQuest Dissertations and Theses. https://academiccommons.columbia.edu/doi/10.7916/D88058PP

Fadjo, C. L. (2012b). *Developing computational thinking through grounded embodied cognition* (Publication Number 3506300) [Ph.D., Columbia University]. ProQuest Dissertations & Theses A&I.

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning*, *32*(6), 576-593. https://doi.org/10.1111/jcal.12155

Flannery, L. P., & Bers, M. U. (2013). Let's Dance the "Robot Hokey-Pokey!". *Journal of Research on Technology in Education*, *46*(1), 81-101. 10.1080/15391523.2013.10782614

García-Valcárcel-Muñoz-Repiso, A., & Caballero-González, Y.-A. (2019). Robotics to develop computational thinking in early Childhood Education. *Comunicar. Media Education Research Journal*, *27*(1), 1-14. https://doi.org/10.3916/C59-2019-06

Georgiou, K., & Angeli, C. (2019). Developing preschool children's computational thinking with educational robotics: The role of cognitive differences and scaffolding. The 16th

International Conference on Cognition and Exploratory Learning in the Digital Age, Cagliari, Italy.

Gerosa, A., Koleszar, V., Tejera, G., Gómez-Sena, L., & Carboni, A. (2021). Cognitive abilities and computational thinking at age 5: Evidence for associations to sequencing and symbolic number comparison. *Computers and Education Open*, *2*, 100043. https://doi.org/10.1016/j.caeo.2021.100043

Gibson, J. P. (2012). *Teaching graph algorithms to children of all ages* Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education, Haifa, Israel. https://doi-org.ezproxy.eduhk.hk/10.1145/2325296.2325308

Gordon, M., Rivera, E., Ackermann, E., & Breazeal, C. (2015). *Designing a relational social robot toolkit for preschool children to explore computational concepts* Proceedings of the 14th International Conference on Interaction Design and Children, Boston, Massachusetts. https://doi.org/10.1145/2771839.2771915

Gözüm, A. İ. C., Papadakis, S., & Kalogiannakis, M. (2022). Preschool teachers' STEM pedagogical content knowledge: A comparative study of teachers in Greece and Turkey. *Frontiers in Psychology*, *13*, 996338. https://doi.org/10.3389/fpsyg.2022.996338

Grossman, P. L., & Richert, A. E. (1988). Unacknowledged knowledge growth: A re-examination of the effects of teacher education. *Teaching and Teacher Education*, *4*(1), 53-62. https://doi.org/10.1016/0742-051X(88)90024-8

Haines, S., Krach, M., Pustaka, A., Li, Q., & Richman, L. (2019). The effects of computational thinking professional development on STEM teachers' perceptions and pedagogical practices. *Athens Journal of Sciences*, *6*(2), 97-122. https://doi.org/10.30958/ajs.6-2-2

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

Huang, W., & Looi, C.-K. (2021). A critical review of literature on "unplugged" pedagogies in K-12 computer science and computational thinking education. *Computer Science Education*, *31*(1), 83-111. https://doi.org/10.1080/08993408.2020.1789411

International Society for Technology in Education, I. (2011). *In Operational definition of computational thinking for K-12 education*. Retrieved from https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Israel-Fishelson, R., & Hershkovitz, A. (2022). Studying interrelations of computational thinking and creativity: A scoping review (2011-2020). *Computers & Education*, *176*, Article 104353. https://doi.org/10.1016/j.compedu.2021.104353

Jacobs, J. K., Kawanaka, T., & Stigler, J. W. (1999). Integrating qualitative and quantitative approaches to the analysis of video data on classroom teaching. *International Journal of Educational Research*, *31*(8), 717-724. https://doi.org/10.1016/S0883-0355(99)00036-1

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood. *Early Childhood Education Journal*, *41*(4), 245-255. https://doi.org/10.1007/s10643-012-0554-5

Khoo, K. Y. (2020). A case study on how children develop computational thinking collaboratively with robotics toys. *International Journal of Educational Technology and Learning*, *9*, 2020. https://doi.org/10.20448/2003.91.39.51

Kite, V., Park, S., & Wiebe, E. (2021). The Code-Centric Nature of Computational Thinking

Education: A Review of Trends and Issues in Computational Thinking Education Research. *Sage Open*, *11*(2), Article 21582440211016418. https://doi.org/10.1177/21582440211016418

Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. *Improving schools*, *19*(3), 267-277. 10.1177/1365480216659733

Kong, S.-C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education*, *3*(4), 377-394. https://doi.org/10.1007/s40692-016-0076-z

Krauss, S., Brunner, M., Kunter, M., Baumert, J., Blum, W., Neubrand, M., & Jordan, A. (2008). Pedagogical content knowledge and content knowledge of secondary mathematics teachers. *Journal of Educational Psychology*, *100*(3), 716. https://doi.org/10.1037/0022-0663.100.3.716

Lee, J., & Junoh, J. (2019). Implementing unplugged coding activities in early childhood classrooms. *Early Childhood Education Journal*, *47*(6), 709-716. https://doi.org/10.1007/s10643-019-00967-z

Lee, S. J., Francom, G. M., & Nuatomue, J. (2022). Computer science education and K-12 students' computational thinking: A systematic review. *International Journal of Educational Research*, *114*, 102008. https://doi.org/10.1016/j.ijer.2022.102008

Lee, T. Y., Mauriello, M. L., Ahn, J., & Bederson, B. B. (2014). CTArcade: Computational thinking with games in school age children. *International Journal of Child-Computer Interaction*, *2*(1), 26-33. https://doi.org/10.1016/j.ijcci.2014.06.003

Li, W., & Yang, W. (2023). Promoting children's computational thinking: A quasi-experimental study of web-mediated parent education. *Journal of Computer Assisted Learning*. https://doi.org/10.1111/jcal.12818

Liberati, A., Altman, D. G., Tetzlaff, J., Mulrow, C., Gøtzsche, P. C., Ioannidis, J. P. A., Clarke, M., Devereaux, P. J., Kleijnen, J., & Moher, D. (2009). The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration. *Journal of Clinical Epidemiology*, *62*(10), e1-e34. https://doi.org/10.1016/j.jclinepi.2009.06.006

Lu, J. J., & Fletcher, G. H. L. (2009). *Thinking about computational thinking* Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, USA. https://doi.org/10.1145/1508865.1508959

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

Macrides, E., Miliou, O., & Angeli, C. (2021). Programming in early childhood education: A systematic review. *International Journal of Child-Computer Interaction*, 100396. https://doi.org/10.1016/j.ijcci.2021.100396

Macrine, S. L., & Fugate, J. M. (2022). *Movement matters: How embodied cognition informs teaching and learning*. MIT Press.

McCormick, K. I., & Hall, J. A. (2021). Computational thinking learning experiences, outcomes, and research in preschool settings: a scoping review of literature. *Education and Information Technologies*. https://doi.org/10.1007/s10639-021-10765-z

McCray, J. S., & Chen, J.-Q. (2012). Pedagogical Content Knowledge for Preschool Mathematics: Construct Validity of a New Teacher Interview. *Journal of Research in Childhood Education*, *26*(3), 291-307. https://doi.org/10.1080/02568543.2012.685123

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica*, *22*(3),

276-282. https://doi.org/10.1007/s10798-020-09616-8.

Metin, S. (2020). Activity-based unplugged coding during the preschool period. *International Journal of Technology and Design Education*, 1-17. https://doi.org/10.1007/s10798-020-09616-8

Mills, K., Coenraad, M., Ruiz, P., Burke, Q., & J, W. (2021). Computational thinking for an inclusive world: a resource for educators to learn and lead. *Digital Promise*. https://doi.org/10.51388/20.500.12265/138

Moore, T. J., Brophy, S. P., Tank, K. M., Lopez, R. D., Johnston, A. C., Hynes, M. M., & Gajdzik, E. (2020). Multiple Representations in Computational Thinking Tasks: A Clinical Study of Second-Grade Students. *Journal of Science Education and Technology*, *29*(1), 19-34. https://doi.org/10.1007/s10956-020-09812-0

Murcia, K. J., & Tang, K. S. (2019). Exploring the multimodality of young children's coding. . *Australian Educational Computing*, *34*(1). http://journal.acce.edu.au/index.php/AEC/article/view/208

Nam, K. W., Kim, H. J., & Lee, S. (2019). Connecting Plans to Action: The Effects of a Card-Coded Robotics Curriculum and Activities on Korean Kindergartners. *The Asia-Pacific Education Researcher*, *28*(5), 387-397. https://doi.org/10.1007/s40299-019-00438-4

Neuman, S. B., & Cunningham, L. (2009). The impact of professional development and coaching on early language and literacy instructional practices. *American educational research journal*, *46*(2), 532-566. https://doi.org/10.3102/0002831208328088

Newhouse, C. P., Cooper, M., & Cordery, Z. (2017). Programmable toys and free play in early childhood classrooms. *Australian Educational Computing*, *32*(1). http://journal.acce.edu.au/index.php/AEC/article/view/147/pdf

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, *11*(1), 1-17. https://doi.org/10.1080/20004508.2019.1627844

Ogegbo, A. A., & Ramnarain, U. (2021). A systematic review of computational thinking in science classrooms. *Studies in Science Education*, 1-28. https://doi.org/10.1080/03057267.2021.1963580

Otterborn, A., Schönborn, K. J., & Hultén, M. (2020). Investigating preschool educators' implementation of computer programming in their teaching practice. *Early Childhood Education Journal*, *48*(3), 253-262. https://doi.org/10.1007/s10643-019-00976-y

Papadakis, S. (2018). Is pair programming more effective than solo programming for secondary education novice programmers?: A case study. *International Journal of Web-Based Learning and Teaching Technologies*, *13*(1), 1-20. https://doi.org/10.4018/IJWLTT.2018010101

Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. *International Journal of Mobile Learning and Organisation*, *10*(3), 187-202. https://doi.org/10.1504/IJMLO.2016.077867

Perin, D. (2011). Facilitating student learning through contextualization: A review of evidence. *Community College Review*, *39*(3), 268-295. https://doi.org/10.1177/0091552111416227

Pila, S., Aladé, F., Sheehan, K. J., Lauricella, A. R., & Wartella, E. A. (2019). Learning to code via tablet applications: An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children. *Computers & Education*, *128*, 52-62. https://doi.org/10.1016/j.compedu.2018.09.006

Portelance, D. J., Strawhacker, A. L., & Bers, M. U. (2016). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, *26*(4), 489-504. https://doi.org/10.1007/s10798-015-9325-0

Pugnali, A., Sullivan, A., & Bers, M. U. (2017). The impact of user interface on young children's computational thinking. *Journal of Information Technology Education. Innovations in Practice*, *16*, 171-193. https://doi.org/10.28945/3768

Pyle, A., & Danniels, E. (2017). A Continuum of Play-Based Learning: The Role of the Teacher in Play-Based Pedagogy and the Fear of Hijacking Play. *Early Education and Development*, *28*(3), 274-289. https://doi.org/10.1080/10409289.2016.1220771

Qu, J. R., & Fok, P. K. (2021). Cultivating students' computational thinking through student–robot interactions in robotics education. *International Journal of Technology and Design Education*, 1-20. https://doi.org/10.1007/s10798-021-09677-3

Relkin, E., & Bers, M. (2021). Techcheck-k: A measure of computational thinking for kindergarten children. IEEE global engineering education conference, Anchorage, Alaska, USA.

Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology*, *29*(4), 482-498. https://doi.org/10.1007/s10956-020-09831-x

Relkin, E., de Ruiter, L. E., & Bers, M. U. (2021). Learning to code and the acquisition of computational thinking by young children. *Computers & Education*, *169*, 104222. https://doi.org/10.1016/j.compedu.2021.104222

Resnick, M., & Robinson, K. (2017). *Lifelong kindergarten: Cultivating creativity through*

projects, passion, peers, and play. MIT press.

Rijke, W. J., Bollen, L., Eysink, T. H., & Tolboom, J. L. (2018). Computational thinking in primary school: An examination of abstraction and decomposition in different age groups. *Informatics in education*, *17*(1), 77-92. https://infedu.vu.lt/journal/INFEDU/article/55/info

Rojas, R. L. M. (2008). *Pedagogical content knowledge in early childhood: A study of teachers' knowledge* (Publication Number 3313157) [Ph.D., Loyola University Chicago]. Education Database; ProQuest Dissertations & Theses A&I.

Romero, M., Lille, B., Viéville, T., Duflot-Kremer, M., de Smet, C., & Belhassein, D. (2018). Analyse comparative d'une activité d'apprentissage de la programmation en mode branché et débranché. Educode-Conférence internationale sur l'enseignement au numérique et par le numérique,

Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K-12: In-service teacher perceptions of computational thinking. In *Computational thinking in the STEM disciplines* (pp. 151-164). Springer.

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2020). Designing unplugged and plugged activities to cultivate computational thinking: An exploratory study in early childhood education. *The Asia-Pacific Education Researcher*, *29*(1), 55-66. https://doi.org/10.1007/s40299-019-00478-w

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. the 18th annual conference on innovation and technology in computer science education, Canterbury.

Shulman, L. S. (1986). Those Who Understand: Knowledge Growth in Teaching. *Educational Researcher*, *15*(2), 4-14 **https://doi.org/10.3102/0013189X015002004**

Shulman, L. S. (1987). Knowledge and Teaching:Foundations of the New Reform. *Harvard Educational Review*, *57*(1), 1-23. https://doi.org/10.17763/haer.57.1.j463w79r56455411

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

So, H.-J., Jong, M. S.-Y., & Liu, C.-C. (2020). Computational Thinking Education in the Asian Pacific Region. *The Asia-Pacific Education Researcher*, *29*(1), 1-8. https://doi.org/10.1007/s40299-019-00494-w

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education*, *25*(3), 293-319. https://doi.org/10.1007/s10798-014-9287-7

Strawhacker, A., & Bers, M. U. (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

Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*, *28*(2), 347-376. https://doi.org/10.1007/s10798-017-9400-9

Sullivan, A., & Bers, M. U. (2013). Gender differences in kindergarteners' robotics and programming achievement. *International Journal of Technology and Design Education*, *23*(3), 691-702. https://doi.org/10.1007/s10798-012-9210-z

Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, *26*(1), 3-20. https://doi.org/10.1007/s10798-015-9304-5

Sullivan, A., & Bers, M. U. (2018). Dancing robots: integrating art, music, and robotics in Singapore's early childhood centers. *International Journal of Technology and Design Education*, *28*(2), 325-346. https://doi.org/10.1007/s10798-017-9397-0

Sun, L., Hu, L., & Zhou, D. (2021a). Which way of design programming activities is more effective to promote K-12 students' computational thinking skills? A meta-analysis. *Journal of Computer Assisted Learning*, *37*(4), 1048-1062. https://doi.org/10.1111/jcal.12545

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

Sung, W., Ahn, J., & Black, J. B. (2017). Introducing Computational Thinking to Young Learners: Practicing Computational Perspectives Through Embodiment in Mathematics Education. *Technology, Knowledge and Learning*, *22*(3), 443-463. https://doi.org/10.1007/s10758-017-9328-x

Sung, W., & Black, J. B. (2021). Factors to consider when designing effective learning: Infusing computational thinking in mathematics to support thinking-doing. *Journal of Research on Technology in Education*, *53*(4), 404-426. https://doi.org/10.1080/15391523.2020.1784066

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, *148*, 103798.

https://doi.org/10.1016/j.compedu.2019.103798

Terroba, M., Ribera, J. M., Lapresa, D., & Anguera, M. T. (2021). Education intervention using a ground robot with programmed directional controls: observational analysis of the development of computational thinking in early childhood education. *Revista de Psicodidáctica (English ed.)*, *26*(2), 143-151. https://doi.org/10.1016/j.psicoe.2021.03.002

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, *162*, 104083. https://doi.org/10.1016/j.compedu.2020.104083

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4), 715-728. https://doi.org/10.1007/s10639-015-9412-6

Wang, C. Z., Shen, J., & Chao, J. (2021). Integrating Computational Thinking in STEM Education: A Literature Review. *International Journal of Science and Mathematics Education*. https://doi.org/10.1007/s10763-021-10227-5

Wang, X. C., Choi, Y., Benson, K., Eggleston, C., & Weber, D. (2020). Teacher's Role in Fostering Preschoolers' Computational Thinking: An Exploratory Case Study. *Early Education and Development*, *32*(1), 26-48. https://doi.org/10.1080/10409289.2020.1759012

Wing, J. (2010). *Computational thinking: What and why?* Computer Science Department, Carnegie Mellon University, Pittsburgh, PA https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

https://dl.acm.org/doi/fullHtml/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

Wing, J. M. (2011). Research notebook: Computational thinking—What and why. *The link magazine*, *6*, 20-23. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Wong, G. K. W., & Jiang, S. (2018, 4-7 Dec. 2018). Computational Thinking Education for Children: Algorithmic Thinking and Debugging. 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), NSW, Australia

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.

Yang, W., Ng, D. T. K., & Gao, H. (2022). Robot programming versus block play in early childhood education: Effects on computational thinking, sequencing ability, and self-regulation. *British Journal of Educational Technology*, 1-25. https://doi.org/10.1111/bjet.13215

Yang, W., Ng, D. T. K., & Su, J. (2023). The impact of story-inspired programming on preschool children's computational thinking: A multi-group experiment. *Thinking Skills and Creativity*, *47*, 101218. https://doi.org/10.1016/j.tsc.2022.101218

Yin, R. K. (2009). *Case study research: Design and methods* (Vol. 5). sage.

Yu, J., & Roque, R. (2019). A review of computational toys and kits for young children. *International Journal of Child-Computer Interaction*, *21*, 17-36. https://doi.org/10.1016/j.ijcci.2019.04.001

Zapata-C, M., x00E, ceres, Mart, E., x00Ed, n, B., Rom, M., x00E, n, G., x00E, & lez. (2021). Collaborative Game-Based Environment and Assessment Tool for Learning Computational Thinking in Primary School: A Case Study. *IEEE Transactions on Learning Technologies*, *14*(5), 576-589. https://doi.org/10.1109/TLT.2021.3111108

Zeng, Y., Yang, W., & Bautista, A. (2023a). Computational thinking in early childhood education: Reviewing the literature and redeveloping the three-dimensional framework. *Educational Research Review*, *39*, 100520. https://doi.org/10.1016/j.edurev.2023.100520

Zeng, Y., Yang, W., & Bautista, A. (2023b). Teaching programming and computational thinking in early childhood education: a case study of content knowledge and pedagogical knowledge. *Frontiers in Psychology*, *14*, 1-13. https://doi.org/10.3389/fpsyg.2023.1252718

Zhang, L. C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, *141*, Article 103607. https://doi.org/10.1016/j.compedu.2019.103607

Zhang, X., Chen, Y., Li, D., Hu, L., Hwang, G.-J., & Tu, Y.-F. (2023). Engaging Young Students in Effective Robotics Education: An Embodied Learning-Based Computer Programming Approach. *Journal of Educational Computing Research*, 07356331231213548. https://doi.org/10.1177/07356331231213548

Zhang, Y. (2015). *Pedagogical content knowledge in early mathematics: What teachers know and how it associates with teaching and learning* (Publication Number 3713666) [Ph.D., Loyola University Chicago]. ProQuest Dissertations & Theses A&I.

Zhang, Y. J., Luo, R. H., Zhu, Y. J., & Yin, Y. (2021). Educational Robots Improve K-12 Students' Computational Thinking and STEM Attitudes: Systematic Review. *Journal*

of Educational Computing Research, *59*(7), 1450-1481, Article 0735633121994070. https://doi.org/10.1177/0735633121994070

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research*, *53*(4), 562-590. https://doi.org/10.1177/0735633115608444

Appendix A. Appendix of Study 1

Appendix A-1

Search Engine, Search Term, Date of the Search Execution, Number of Items Found and Additional Information

Search Engine	Search term	Note	Date	Nr of items
Web of Science	("Computational Thinking") AND (preschool* OR kindergarten* OR pre-K* OR prekindergarten OR "early child*" "early age*" OR "early years" OR "young child*" OR "young learners" OR child* OR "elementary education" OR "lower education" OR "primary education" OR "pre-primary education")	Search in topic; Document type: article+conference paper+early access	01.10.2021	119
SCOPUS	TITLE-ABS-KEY ("Computational Thinking") AND TITLE-ABS-KEY(preschool* OR	Search in title, abstract and keywords；	01.10.2021	96

Search Engine	Search term	Note	Date	Nr of items
	kindergarten* OR pre-K* OR prekindergarten OR "early child*" "early age*" OR "early years" OR "young child*" OR "young learners" OR child* OR "elementary education" OR "lower education" OR "primary education" OR "pre-primary education")	Document type: article+conference paper+book		
ERIC	("Computational Thinking") AND (preschool* OR kindergarten* OR pre-K* OR prekindergarten OR "early child*" "early age*" OR "early years" OR "young child*" OR "young learners" OR child* OR "elementary education" OR "lower education" OR "primary education" OR "pre-	Search in title, abstract and identifiers； Document type: academic journal+report+book+dissertation	03.10.2021	87

Search Engine	Search term	Note	Date	Nr of items
	primary education”)			
ScienceDirect	Term 1: (“Computational Thinking”) AND (preschool OR preschooler OR kindergarten OR kindergartner OR pre-K OR prekindergarten) Term 2: (“Computational Thinking”) AND (“elementary education” OR “lower education” OR “primary education” OR “pre-primary education” OR “early years” OR “early age” OR “early child” OR “early childhood”) Term 3: (“Computational Thinking”) AND (“young child” OR“young children”OR “young learners” OR child OR children OR childhood)	Search in title, abstract and Keywords; ScienceDirect allows maximum eight boolean terms in the search term, so we split the search term in three to cover all the words identified as relevant for the search; For wildcards’*’ are not supported by ScienceDirect, we added some extension words. Document type: Review articles, research articles (The literatures in this database	01.10.2021	8+24+37

Search Engine	Search term	Note	Date	Nr of items
		are all academic ones)		
ProQuest	TI, AB, IF((“Computational Thinking”) AND (preschool* OR kindergarten* OR pre-K* OR prekindergarten OR (“early child” OR “early childcare” OR “early childhood” OR “early children”) (“early age” OR “early ages”) OR “early years” OR (“young child” OR “young childhood” OR “young children”) OR “young learners” OR child* OR “elementary education” OR “lower education” OR “primary education” OR “pre-primary education”))	Search in title, abstract and identifiers； Document type: academic journals, dissertations, conference papers, research manuscripts, books	01.10.2021	252

Appendix A-2

The Snowballing Seeds of Literature Search

First round	Publication Date	Journal
Assessing computational thinking: A systematic review of empirical studies	2020	Computers & Education
Preschool children, robots, and computational thinking: A systematic review	2021	International Journal of Child-Computer Interaction
Computational thinking learning experiences, outcomes, and research in preschool settings: a scoping review of literature	2021	Education and Information Technologies
A systematic literature review regarding computational thinking and programming in early childhood education	2021	Education and Information Technologies

Appendix A-3

Coding Framework: The Three-Dimensional CT Framework (Brennan & Resnick, 2012)

CT concepts	Description	CT practices	Description	CT perspectives	Description
Sequence	A set of ordered steps for performing a task	Being iterative and incremental	Problem-solving is an iterative process, with plans being revised step-by-step	Expressing	Regarding computation as a way to create and self-express
Loops	Repetition of the same instruction multiple times	Testing and debugging	Finding and fixing errors	Connecting	Recognizing the value of creating with and for others
Parallelism	Simultaneous running of multiple instructions	Reusing and remixing	Building reusable instructions or new products based on others' work	Questioning	Feeling empowered to raise questions about technology and use it
Events	"One thing causing another thing to happen" (p.4)	Abstracting and modularizing	Extract the basic elements and patterns of complex systems	Others	Other CT perspectives not included in the three-dimensional CT framework

CT concepts	Description	CT practices	Description	CT perspectives	Description
Conditionals	"The ability to make decisions based on certain conditions" (p. 5)	Others	Other CT practices that not included in the three-dimensional CT framework		
Operators	Mathematical and string operations				
Data	"Storing, retrieving, and updating values" (p. 5)				
Others	Other CT concepts that not included in the three-dimensional CT framework				

Appendix A-4

Overview of the Included Studies

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Ahn et al. (2021)	USA	7-9 years	59	QUANT: between-participants designs (posttest-only control design)	A paper-based debugging test	Maze	Three 50-min weekly sessions
Angeli and Valanides (2020)	Southern European Country	5-6 years	50	QUANT: between-participants designs (pretest-posttest control-group design)	A self-developed task-based CT rubric	Bee-Bot	Two 40-min sessions
Bers et al. (2014)	USA	4.9-6.5 years	53		A robot and/or program		

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Bers et al. (2019)	Spain	3-5 years	172	QUANT: one-group, with assessment continuously or after each session MIXED: students' projects analysis, classroom observations, teacher interviews, diary journal, and questionnaire	evaluation Likert scale Solve-Its task-based assessment, PTD Checklist	LEGO robotics kit with CHERP KIBO	Six 60-min to 90-min, 20 h total in 6 weeks 3-5 sessions ranging from 45-min to 75-min in 2-3 weeks

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Cho and Lee (2017)	UK	5-6 years	12	QUANT: one-group posttest-only design	A student self-evaluation scale	LEGO mindstorm NXT	5 50-min sessions for 5 weeks
Clarke-Midura et al. (2021)	USA	kindergarten-aged children	89	QUANT: assessment development	N/A	A Coding Robot	N/A
Critten et al. (2022)	UK	2-4 years	15	QUANT: children's evaluation by adults after each session	A Coding Development Test 3-6 developed by Marinus et al. (2018), Children communication checklist	Bee-Bot	Six 45-60 min sessions

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
del Olmo-Muñoz et al. (2020)	Spain	7-8 years	84	QUANT: a quasi-experimental design (pre-, mid-, and post test)	An instrument consisting of 10 items from the "International Bebras Contest"	Code. Org	Eight 45-min sessions in 8 weeks
Dietz et al. (2019)	USA	Study 1: 4-7 years Study 2: 3-5 years	Study 1: 112 Study 2: 78	QUANT: experiment	N/A	Blocks	10-min task
Ehsan et al. (2020)	USA	5-7 years	10	QUAL: videotaped classroom observations, teacher interviews	N/A	Playground's Big Blue Blocks	30-min engineering design task

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Elkin et al. (2016)	USA	3-5 years	64	QUANT: one-group posttest-only design	Solve-Its task-based assessment	KIBO	Six 90-min sessions in 6 days
Flannery and Bers (2014)	USA	4.4-6.6 years	29	QUANT: one-group pretest-posttest design	"Hokey-Pokey" program completeness assessment rubric	TangibleK robotics kit with CHERP	3 individual sessions
Georgiou and Angeli (2019)	Cyprus	5-6 years	180	QUANT: between-participants designs	Self-developed CT evaluation rubrics	Bee-Bot	N/A
Gerosa (2021)	Uruguay	5-6 years	102	QUANT: cross-sectional correlational design	A CT assessment adapted from Tran's CT questionnaire (Tran, 2019)	RoboTito	Elevn 25-30 min sessions

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Gordon et al. (2015)	USA	4-6 years	22	MIXED: pre-post interview	N/A	SoRo toolkit (Social Robot Toolkit)	30 min
Kazakoff et al. (2013)	USA	4-6 years	27	QUANT: quasi-experimental design (pretest-posttest)	Picture sequencing cards created by Baron-Cohen et al. (1986)	LEGO WeDo robotics with CHERP	1 week
Khoo (2020)	Hong Kong	5 years	3	QUAL: (multiple-case study) classroom observations, teacher interviews, students' artifacts analysis	A picture-story sequencing task	Colby Mouse and Ozobot Bit	Nine 15-min activities

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Metin (2020)	Turkey	5 years	24	QUANT: one-group pretest-posttest design	The Basic Coding Skills Observation Form, the Basic Robotic Coding Skills Observation Form	Cubetto	60-90 min of daily training over 8 days
Moore et al. (2020)	Midwest	7-8 years	3	QUAL: a task-based interview; audio and video recording	Task-based interview assessment	Code and Go™ Robot; Mouse Coding Activity Set	Two 1-hr weekly sessions

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Muñoz-Repiso and Caballero-González (2019)	Spain	3-6 years	131	QUANT: quasi-experiment (pretest-posttest) QUAL: researcher site visits, educator classroom observations, shared collegial reflection and review of educator generated learning stories	The "SSS" rubric used in the TangibleK program (Bers, 2010)	Bee-Bot	7 sessions (duration unclear)
Murcia and Tang (2019)	Australia	3-4 years	8		N/A	Cubetto	6 months

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Nam et al. (2019)	The Republic of Korea	5-6 years	53	QUANT: quasi-experiment (pretest-posttest)	Story sequencing test, Korean version (Ryu, 2003) of Ward's (1993) original problem-solving performance instrument	TurtleBot	90 minutes/ 8 sessions with 12 activities in 8 weeks
Newhouse et al. (2017)	Australia	4-6 years	50	QUAL: classroom observations, teacher interview	Checklist of behaviours drawing upon Bird and Edwards (2014)	Bee-Bot and Sphero robots	6 weeks

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Papadakis et al. (2016)	Greece	5 years	43	QUANT: one-group posttest-only design	Tas-based paper format assessment	ScratchJr	13 h in 6 weeks, two 60-min semiweekly sessions
Pila et al. (2019)	USA	4-5 years	28	MIXED: pre-post children interview, pre-post gameplay assessment	A coding scheme adapted from Bers et al. (2014)	Daisy the Dinosaur and Kodable	five 3-hr sessions in 1 week
Portelance et al. (2016)	USA	5-8 years	62	QUANT: students' projects analysis	Project based assessment	ScratchJr	Twelve 1-hr lessons,

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
							twice a week
Pugnali et al. (2017)	USA	4-7 years	28	MIXED: quasi-experiment (posttest); observations	Solve-Its task-based assessment, PTD checklist	KIBO or ScratchJr	15 h in 1 weeks
Qu and Fok (2021)	China	7-9 years	32	MIXED: rubric scoring, teacher interviews and classroom observations	CT rubric designed by Leonard et al. (2016)	KAZI EV5 and Scratch	Twelve 90-min lessons, thrice weekly

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Relkin et al. (2021)	USA	6-8 years	The experimental group (N=667) The control group (N=181)	QUANT: A quasi-experimental longitudinal design	TechCheck	KIBO	12-15 h in 6-7 weeks
Rijke et al. (2018)	Netherlands	6-12 years	200	QUANT: intervention in different age groups with post test	The number of cards a student could get their partner to guess right (abstraction assessment) and the	Unplugged	Two unplugged lessons

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
					number of decomposed movements (decomposition assessment)		
Saxena et al. (2020)	Hong Kong	3-6 years	11	MIXED: one-group posttest-only design, classroom observations, teacher interviews	A robot and/or program evaluation Likert scale designed by Bers et al. (2014)	Bee-Bot	10 h
Strawhacker and Bers (2015)	USA	5-6 years	35	MIXED: classroom observations and quantitative mid- and post-test assessments	Solve-Its task-based assessment	LEGO WeDo 2.0 with CHERP	13 h lessons in 9 weeks

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Strawhacker and Bers (2019)	USA	5-8 years	57	QUANT: one-group posttest-only design	Solve-Its task-based assessment	ScratchJr	6 weeks
Strawhacker et al. (2018)	USA	5-7 years	6 teachers and 222 students	MIXED: one-group posttest-only design, journal entries and surveys	Solve-Its task-based assessment	ScratchJr	A minimum of 2 lessons and a maximum of 7 lessons
Sullivan and Bers (2013)	USA	Kindergarteners	53	QUANT: one-group, with assessment continuously or after each session	A robot and/or program evaluation Likert scale	RCX brick with CHERP	20 hours in 6 lessons

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Sullivan and Bers (2016)	USA	4-8	60	QUANT: one-group posttest-only design	Robot Parts test, Solve-Its task-based assessment	KIWI robotics with CHERP	8 h in 8 weeks
Sullivan and Bers (2018)	Singapore	3-6 years	98	MIXED: one-group midtest-posttest design, classroom observations, teacher interviews and journals	Solve-Its task-based assessment, PTD Checklists	KIBO	7 h in 7 weeks
Sullivan et al. (2013)	USA	5 years	37	QUAL: teacher interviews, videos, photographs, and classroom observations	N/A	LEGO WeDo robotics with CHERP	10 h over 5 days

Authors (Year)	Region/Country	Participants' age	Sample size	Research method	CT measurement(s)	CT Intervention tool(s)	CT Intervention duration
Sung and Black (2021)	USA	7-9 years	115	QUANT: A 2 × 2 factorial design experiment	A paper-based programing skill test	Hopscotch	Six 50-min sessions
Sung et al. (2017)	USA	5-7 years	66	QUANT: A 2 × 2 factorial design experiment	Programming task assessment with a rubric	ScratchJr	Five 1-hr sessions
Terroba et al. (2021)	Spain	5 years	24	MIXED: classroom observations, quasi-experiment	A self-developed CT behavior observation system	A ground robot	10-12 hours
Wang et al. (2020)	USA	3-4 years	3	QUAL: videotaped classroom observations	N/A	Code-a-pillar	12 weekly 20-min sessions

Appendix A-5

CT Concepts Emphasized by Each Study

Study	CT concepts							
	Sequences	**Events**	**Loops**	**Parallelism**	**Conditionals**	**Operators**	**Data**	**Others**
Ahn et al. (2021)	√							
Angeli and Valanides (2020)	√	√						
Bers et al. (2014)	√	√	√		√			Control flow
Bers et al. (2019)	√	√	√		√			
Cho and Lee (2017)								
Clarke-Midura et al. (2021)								
Critten et al. (2022)	√							Representation
del Olmo-Muñoz et al. (2020)	√	√	√					Representation
Dietz et al. (2019)								
Ehsan et al. (2020)								

Study	CT concepts							
	Sequences	**Events**	**Loops**	**Parallelism**	**Conditionals**	**Operators**	**Data**	**Others**
Elkin et al. (2016)	√	√	√					Hardware/Software
Flannery and Bers (2014)	√	√						
Georgiou and Angeli (2019)	√							
Gerosa (2021)	√		√		√			
Gordon et al. (2015)	√	√		√				
Kazakoff et al. (2013)	√							
Khoo (2020)								Automation Representation
Metin (2020)	√							Representation
Moore et al. (2020)	√							Representation
Muñoz-Repiso and Caballero-González (2019)	√	√						Representation

Study	CT concepts							
	Sequences	**Events**	**Loops**	**Parallelism**	**Conditionals**	**Operators**	**Data**	**Others**
Murcia and Tang (2019)	√	√	√					Representation
Nam et al. (2019)	√							Representation
Newhouse et al. (2017)								
Papadakis et al. (2016)	√							
Pila et al. (2019)	√		√		√			
Portelance et al. (2016)	√	√	√					Control flow
Pugnali et al. (2017)	√		√		√			
Qu and Fok (2021)		√	√		√			
Relkin et al. (2021)	√	√	√		√			Hardware/Software Control structures Representation
Rijke et al. (2018)								

Study	CT concepts							
	Sequences	**Events**	**Loops**	**Parallelism**	**Conditionals**	**Operators**	**Data**	**Others**
Saxena et al. (2020)	√							
Strawhacker and Bers (2015)	√		√					
Strawhacker and Bers (2019)	√	√	√					
Strawhacker et al. (2018)	√		√					
Sullivan and Bers (2013)	√	√	√		√			Control flow
Sullivan and Bers (2016)		√	√		√			Hardware/Software
Sullivan and Bers (2018)	√	√	√		√			
Sullivan et al. (2013)	√		√					Hardware/Software
Sung and Black (2021)	√							
Sung et al. (2017)	√							
Terroba et al. (2021)								
Wang et al. (2021)								

Study	CT concepts							
	Sequences	**Events**	**Loops**	**Parallelism**	**Conditionals**	**Operators**	**Data**	**Others**
Frequency	31	16	18	1	10	0	0	Representation: 9 Control flow: 3 Control structures: 1 Hardware/Software: 4 Automation: 1

Appendix A-6

CT Practices Emphasized by Each Study

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
Ahn et al. (2021)		√			Pattern recognition
Angeli and Valanides (2020)		√	√ (Decomposition) √ (Abstraction)		Algorithmic design
Bers et al. (2014)	√	√	√ (Decomposition)		
Bers et al. (2019)	√	√			
Cho and Lee (2017)			√		Algorithmic design
Clarke-Midura et al. (2021)		√	√ (Decomposition)		Algorithmic design

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
					Spatial reasoning
Critten et al. (2022)		√			Algorithmic design Logical thinking
del Olmo-Muñoz et al. (2020)		√	√ (Abstraction)		Algorithmic design Pattern recognition Generalization
Dietz et al. (2019)			√ (Decomposition)		
Ehsan et al. (2020)		√	√ (Decomposition)		Algorithmic design Pattern recognition Simulations

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
Elkin et al. (2016)	√	√			
Flannery and Bers (2014)					
Georgiou and Angeli (2019)		√			Algorithmic design
Gerosa (2021)		√			Algorithmic design
Gordon et al. (2015)		√			
Kazakoff et al. (2013)					
Khoo (2020)			√ (Decomposition) √ (Abstraction)		Algorithmic design
Metin (2020)					

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
Moore et al. (2020)		√	√ (Decomposition) √ (Abstraction)		Algorithmic design Pattern recognition
Muñoz-Repiso and Caballero-González (2019)		√			
Murcia and Tang (2019)		√	√ (Decomposition)		
Nam et al. (2019)	√				
Newhouse et al. (2017)		√			

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
Papadakis et al. (2016)					
Pila et al. (2019)					
Portelance et al. (2016)					
Pugnali et al. (2017)		√			
Qu and Fok (2021)		√	√ (Decomposition) √ (Abstraction)		Algorithmic design Generalizing and problem transfer Logical thinking
Relkin et al. (2021)			√ (Decomposition)		Algorithmic design
Rijke et al. (2018)			√ (Decomposition)		

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
			√ (Abstraction)		
Saxena et al. (2020)					Pattern recognition Algorithmic design
Strawhacker and Bers (2015)					
Strawhacker and Bers (2019)					
Strawhacker et al. (2018)					
Sullivan and Bers (2013)	√	√	√ (Decomposition)		
Sullivan and Bers (2016)		√	√ (Decomposition)		
Sullivan and Bers (2018)					

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
Sullivan et al. (2013)	√	√			
Sung and Black (2021)			√ (Decomposition) (Abstraction)		Pattern recognition
Sung et al. (2017)					Pattern recognition
Terroba et al. (2021)		√	√ (Decomposition)		
Wang et al. (2021)		√	√ (Problem reformulation/Decomposition)		
Frequency	6	23	Decomposition: 16 Abstraction: 7	0	Algorithmic design:13 Pattern recognition: 7

Study	CT Practices				
	Being iterative and incremental	**Testing and debugging**	**Abstracting and modularizing**	**Reusing and remixing**	**Others**
					Generalization: 2 Logical thinking: 2 Simulations: 1 Spatial reasoning: 1

Appendix A-7

CT Perspectives Emphasized by Each Study

Study	CT Perspectives			
	Expressing	**Connecting**	**Questioning**	**Others**
Ahn et al. (2021)				
Angeli and Valanides (2020)				
Bers et al. (2014)	√	√		Choices of conduct
Bers et al. (2019)	√	√		Choices of conduct
Cho and Lee (2017)				
Clarke-Midura et al. (2021)				
Critten et al. (2022)		√		
del Olmo-Muñoz et al. (2020)		√		
Dietz et al. (2019)				
Ehsan et al. (2020)				
Elkin et al. (2016)	√	√		
Flannery and Bers (2014)				
Georgiou and Angeli (2019)				

Study	CT Perspectives			
	Expressing	**Connecting**	**Questioning**	**Others**
Gerosa (2021)				
Gordon et al. (2015)				
Kazakoff et al. (2013)				
Khoo (2020)				
Metin (2020)		√		
Moore et al. (2020)				
Muñoz-Repiso and Caballero-González (2019)	√	√		
Murcia and Tang (2019)	√	√		
Nam et al. (2019)				
Newhouse et al. (2017)	√			
Papadakis et al. (2016)				
Pila et al. (2019)				
Portelance et al. (2016)	√			
Pugnali et al. (2017)	√	√		Choices of conduct
Qu and Fok (2021)				
Relkin et al. (2021)	√	√		
Rijke et al. (2018)				
Saxena et al. (2020)				

Study	CT Perspectives			
	Expressing	**Connecting**	**Questioning**	**Others**
Strawhacker and Bers (2015)	√	√		
Strawhacker and Bers (2019)		√		
Strawhacker et al. (2018)				Choices of conduct
Sullivan and Bers (2013)				
Sullivan and Bers (2016)				
Sullivan and Bers (2018)	√	√		Perseverance
Sullivan et al. (2013)	√	√		
Sung and Black (2021)				
Sung et al. (2017)				
Terroba et al. (2021)				
Wang et al. (2021)		√		Perseverance
Frequency	12	15	0	Choices of conduct: 4 Perseverance: 2

Appendix B. Appendix of Study 2

Appendix B-1

Examples of Data Analysis

Data Types	**Transcripts**	**CK indicators involved**	**PK indicators involved**			
			Teaching context	**Activity structure**	**Pedagogical approaches**	**Pedagogical strategies**
Video data	Teacher: Today, Qiqi will take a spaceship to reach the Moon, Jupiter and Uranus to explore the mysteries of the three planets. Qiqi wants to go to the Moon first. [Contextualization] Do you know where the Moon is located? Children: Row 5, column 7. Teacher: Qiqi needs to take a route with loops to reach the Moon. Have	Loops, Representation	Group activity	Highly structured	Task-based learning	Contextualization, External memory support scaffolding, Embodied cognition

	you found a route with loops? [Loops] (Child 1 raises his hand) Teacher: Yes, please share your idea. Child 1: One step forward, one step to the left, one step forward, one step to the left, one step forward, one step to the left (Child 1 describes the route while gesturing with his hand) (The teacher notes down the route described by Child 1 on the board using arrows).[External memory support scaffolding] [Representation] Teacher: Let's move our fingers along the route XXX described and see if it's correct [Embodied cognition]					

	…					
Interview data	Interviewer: What do you consider the core content of early programming and CT, or what do you include in your unplugged programming curriculum? Teacher: In the first semester of our unplugged programming curriculum, children learned how to use programming blocks to give instructions such as “go forward” “go backward” “go left” and “go right” through floor games. [Representation] [Embodied cognition] In the second semester, in addition to learning how to give instructions of walking in	Representation, Sequences, Loops, Conditionals, Expressing and creating				Embodied cognition

	different directions, children also learned how to give instructions for walking several steps in different directions. [Sequences] In the K2 class, we introduce board games as a medium for learning. [Embodied cognition]Children also need to learn about conditionals and loops. [Conditionals and Loops] In the K3 class, the routes children need to program are longer and more complex [Sequences] compared to the K2 classes. Children learn to use a variety of instructions for sequences, conditionals, and loops in a single route. [Sequences, Conditionals, and					

	Loops] They also design different tools on blank Tool Blocks [Expressing and creating] to help Qiqi solve problems. …					
Lesson plan	**Activity 2: Exploring the Planets** **Learning Objectives** 1. To use the Loops Blocks independently and use the correct Number Blocks and Directional Blocks to solve problems. [Sequences, Loops] 2. To experience the joy of cooperative programming. [Connecting]	Sequences, Loops, Algorithmic design, Connecting	Group activity	Highly structured	Task-based learning	Contextualization, Pair programming

	Learning Preparation 1. Scenario Blocks: Moon Block, Jupiter Block, Uranus Block, Meteorite Blocks. 2. Programming Blocks: Directional Blocks, Number Blocks, Loops Block. 3. The Outer Space Board. 4. PPT. **Learning process** **1. <u>Create a situation of going to planets to explore their mysteries.</u>** <u>[Contextualization]</u> --Do you remember Qiqi's dream? (PPT: outer space)					

	-- What equipment does Qiqi need to take with him to explore outer space? (PPT: spacesuit, oxygen kit, and translator) --With these equipments, Qiqi can take a spaceship to explore outer space! Qiqi wants to go to the Moon, Jupiter, and Uranus to explore their mysteries! (PPT: the Moon, Jupiter, and Uranus) **2. Design routes with loops to the Moon** --Qiqi plans to go to the Moon first. --Qiqi has to take a route with loops to reach the Moon. Have you found a route with loops? [Sequences, Loops]					

	(Ask several children to answer) ---You designed different routes with loops to help Qiqi reach the Moon. What is the mystery of the Moon? Let's listen to it. (PPT: the mystery of the Moon) **3. Design routes with loops to Jupiter and Uranus** -- What are the mysteries of Jupiter and Uranus? Do you want to know? -- We have to find these two planets first. Do you know where the two planets are located?? -- Again, Qiqi has to take routes with loops to reach Jupiter and Uranus. Can you help Qiqi design different					

	routes with loops? [Sequences, Loops] --There are many meteorites in outer space. Remember to go around them! [Algorithmic design] (Children using the unplugged coding set in pairs [Pair programming] to design routes with loops to Jupiter and Uranus while the teacher goes around to check and guide them.) (Children share their looping routes to Jupiter and Uranus.) --You designed different routes with loops to help Qiqi reach Jupiter and Uranus. What are the mysteries of Jupiter and Uranus? Let us listen to					

	it. (PPT: the mysteries of Jupiter and Uranus) --With your help, Qiqi has reached the Moon, Jupiter and Uranus. Where else will Qiqi go on the spaceship? See you next time.					

Appendix B-2

Interview Protocol: Teachers' Content Knowledge and Pedagogical Knowledge in Early Programming and CT

Before the interview

Thank you very much for allowing me to observe and videotape your classes this semester and for taking the time to be interviewed. I want to ask you some questions based on my observed activities. Since your answers are important to my research, I would like to record our conversation, okay?

Start the audio recording with the consent of the interviewee

(The following questions are only the outline of the interview, and the actual interview will be flexible according to the teacher's answers)

Part A Basic Information

1. Could you briefly introduce yourself, including your age, education, working experience, etc.?
2. How many years of early childhood education experience do you have in total (excluding years of study)?
3. How long have you taught programming and CT?

Part B Content Knowledge

4. What do you think is the core content of early programming and CT?
5. There are 12 programming and CT activities for this semester, and here are the lesson plans for these 12 activities (show the lesson plans). Can you tell me the core content covered in each activity?

6. (If “decomposition” is not mentioned) What could the children learn from the “Backward Inference Task”?

7. How do you understand XXX (XXX stands for the core content mentioned by Ms. Wu)?

Part C Pedagogical Knowledge

8. What materials did you provide to help children learn programming and CT? Why did you provide these materials?

9. Have you conducted other forms of programming and CT activities besides group activities (such as integrating programming and CT into the learning center, children’s daily routines or other learning domains?)

i. If yes, how?

ii. If no, do you have any ideas about how to integrate?

10. What is the basic process of the programming and CT group activities?

11. What pedagogical approaches did you employ (e.g., task-based learning, play-based learning, project-based learning)? Why did you employ this approach?

12. What pedagogical strategies did you use in teaching programming and CT?

13. Why did you use XXX (XXX stands for the pedagogical strategies mentioned by Ms. Wu)?

14. What do you think you did and did not do well in supporting young children to learn programming and CT? Why?

Part D Final Question of the Interview

15. Do you have any further comments?

Appendix B-3

Steps for Making an Unplugged, Boardgame-Like Coding Set

Step 1: Create the object to be programmed. Cut out a card (being careful that the card size does not exceed the size of the grid on the board) and draw a pawn on the card, or use a toy as the object to be programmed.

Step 2: Make a grid map for the pawn to move. Take a large piece of paper and draw grids on it, for example, 10 by 10 grids.

Step 3: Create chess pieces for programming tasks. Cut some cards (being careful that the size of the cards is at most the size of the grid on the board) and draw places and tools that appear in the programming tasks on them. Use your imagination to create fun scenarios. When playing, place these cards on the grid map according to the designed programming task.

Step 4: Make programming cards. Cut some cards and write numbers, arrows, and patterns that represent loops and conditionals on them to make number cards, directional cards, loops cards, and conditional instruction cards.

Then you can play the board game with your friends! One person designs a programming task, one "writes" instructions by placing programming cards on the paper or floor, and one moves the pawn on the grid map to verify the instructions. You can also make up other rules to make the game more enjoyable!

Appendix C. Appendix of Study 3

Programming activities	The content of programming activities	The objective of programming activities	Examples of programming tasks
Activity 1	Hardware and software, events, sequences (1)	1. Learn about the components of the MOBLO programming kit. 2. Learn the basic operations of connecting the Sensor Board to a tablet, arranging electronic blocks, etc. 3. Comprehend the concept of "sequence". 4. Recognize the "forward," "backward," "left," and "right" Direction Blocks and be able to arrange them in a specific sequence to create programs that help Kobe complete simple	

		tasks.	
Activity 2	Sequences (2), representation, decomposition (1)	1. Consolidate the concept of "sequence". 2. Be able to arrange Direction Blocks in a specific sequence to create programs that help Kobe complete more complex tasks. 3. Understand the concept of "representation" by observing symbols in the Programming Instructions Record Bar. 4. Decompose complex problems into smaller, manageable parts during the program creation process.	

Activity 3	Sequences (3), algorithmic design	1. Consolidate the concept of "sequence". 2. Be able to arrange direction and action blocks in a specific sequence to create programs that help Kobe complete complex tasks. 3. Design a series of ordered steps or actions to solve problems.	
Activity 4	Loops (1), pattern recognition (1)	1. Understand the concept of "loop". 2. Recognize NFC Blocks and Number Cards and learn how to use Direction Blocks, NFC blocks, and Number Cards to create loop instructions. 3. Identify loop units and loop counts in a simple route and create loop instructions using	

		Direction Blocks, NFC Blocks, and Number Cards.	
Activity 5	Loops (2), pattern recognition (2), debugging (1)	1. Consolidate the concept of "loop". 2. Identify loop units and loop counts in a more complex route and create loop instructions using Direction Blocks, NFC Blocks, and Number Cards. 3. When encountering errors in the program, apply different methods to locate and correct the errors.	

Activity 6	Loops (3), pattern recognition (3), debugging (2)	1. Consolidate the concept of "loop". 2. Identify loop units and loop counts in a more complex route and create loop instructions using Direction Blocks, NFC Blocks, and Number Cards. 3. When encountering errors in the program, apply different methods to locate and correct the errors.	
Activity 7	Loops (4), pattern recognition (4), debugging (3)	Same as Activity 6	

Activity 8	Loops (5), pattern recognition (5), debugging (4)	Same as Activity 6	
Activity 9	Loops (6), pattern recognition (6), debugging (5)	Same as Activity 6	

Activity 10	Conditionals (1), decomposition (2)	1. Understand the concept of "conditionals". 2. Correctly use Direction Blocks, Action Blocks, NFC Blocks, and Tool Cards to create conditional instructions and solve simple problems by making choices based on the situation. 3. During the program creation process, decompose complex problems into smaller, manageable parts.	

Activity 11	Conditionals (2), decomposition (3)	Same as Activity 10	
Activity 12	Conditionals (3), decomposition (4)	Same as Activity 10	

Note: "The content of programming activities" column in this table only provides a list of the primary teaching content for each activity. Each programming activity integrates hardware and software, events, representations, algorithmic design, and debugging.

Appendix D. The Ethical Approval

香港教育大學
The Education University of Hong Kong

14 November 2022

Ms ZENG Yue
Doctor of Education Programme
Graduate School

Dear Ms Zeng,

Application for Ethical Review <Ref. no. 2021-2022-0315>

I am pleased to inform you that approval has been given by the Human Research Ethics Committee (HREC) for your research project:

Project title: Effects of Plugged and Unplugged Programming Curricula on Computational Thinking, Self-Regulation, Flow Experience and Self-Efficacy of Children Aged 5-6

Ethical approval is granted for the project period from 14 November 2022 to 30 December 2023. If a project extension is applied for lasting more than 3 months, HREC should be contacted with information regarding the nature of and the reason for the extension. If any substantial changes have been made to the project, a new HREC application will be required.

Please note that you are responsible for informing the HREC in advance of any proposed substantive changes to the research proposal or procedures which may affect the validity of this ethical approval. You will receive separate notification should a fresh approval be required.

Thank you for your kind attention and we wish you well with your research.

Yours sincerely,

[redacted]

Patsy Chung (Ms)
Secretary
Human Research Ethics Committee

c.c. Prof CHOU Kee Lee, Chairperson, Human Research Ethics Committee

香港新界大埔露屏路十號
10 Lo Ping Road, Tai Po, New Territories, Hong Kong
T (852) 2948 8888 F (852) 2948 6000 www.eduhk.hk

Appendix E. Consent Forms (English and Chinese Versions)

Consent Form and Information Sheet for PARENTS

THE EDUCATION UNIVERSITY OF HONG KONG

Department of Early Childhood Education

CONSENT TO PARTICIPATE IN RESEARCH

Effects of Plugged and Unplugged Programming Curricula on Computational Thinking, Self-Regulation, Flow Experience and Self-Efficacy of Children Aged 5-6

I _____ hereby consent to my child participating in a project supervised by Dr. Weipeng Yang and Dr. Alfredo Bautista and conducted by Yue Zeng, who are staff / students of the department of Early Childhood Education in The Education University of Hong Kong.

I understand that information obtained from this research may be used in future research and may be published. However, our right to privacy will be retained, i.e., the personal details of my child will not be revealed.

The procedure as set out in the **attached** information sheet has been fully explained. I

understand the benefits and risks involved. My child's participation in the project is voluntary.

I acknowledge that we have the right to question any part of the procedure and can withdraw at any time without negative consequences.

Name of participant _____

Name of Parent or Guardian _____

Signature of Parent or Guardian _____

Date _____

INFORMATION SHEET

Effects of Plugged and Unplugged Programming Curricula on Computational Thinking, Self-Regulation, Flow Experience and Self-Efficacy of Children Aged 5-6

Your children are invited to participate in a project supervised by Dr. Weipeng Yang and Dr. Alfredo Bautista and conducted by Yue Zeng, who are staff / students of the department of Early Childhood Education in The Education University of Hong Kong.

The introduction of the research

A) What does the research involve?

- I will design a series of plug-in and unplugged programming activities for children to help them learn sequencing, loops, conditionals, decomposition, debugging, and other programming concepts and skills. Unplugged programming curriculum refers to teaching programming without digital devices and often involves paper and pencil, cards, sticker books, as well as body movements, while plugged programming curriculum refers to teaching programming with the use of digital devices. I will use MBOLO in the plugged programming group and use unplugged materials in the unplugged programming group.

- I will provide training to teachers on what computational thinking is and how to develop it in young children; and

- I will evaluate the impact of plugged and unplugged programming courses on children's computational thinking, self-regulation skills, flow experiences, and programming self-efficacy.

- I will videotape the programming activities (about two months) carried out by the two experimental classes.

- After all programming activities, interviews will be conducted with two teachers and several children from the experimental classes, and all interviews will be recorded.

The methodology of the research

A) Procedure of the research

- Each child's computational thinking will be assessed (pre-test and post-test) before and after the programming curriculum (intervention). The computational thinking assessment typically takes 12 minutes to administer to young children and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's self-regulation skills will be assessed (pre-test and post-test) before and after the programming curriculum (intervention). The self-regulation assessment usually takes 15 minutes to administer to young children and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's flow experience will be assessed after the programming program (intervention). The assessment will take about two minutes and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's programming self-efficacy will be assessed after the programming program (intervention). The programming self-efficacy assessment will be completed by the class teacher.

- Focus group interviews will be conducted with respectively ten children from each experimental group. The interviews will be videotaped and will last about one hour.

- The teacher will conduct the programming activities (the details of each programming activity see Table 1) and the researcher will videotaped all the programming activities (12 sessions, 40 mins per session).

Table 1

Programming Courses

Session	**Plugged activities**		**Unplugged activities**	
	Activity	**Objective**	**Activity**	**Objective**
Session 1	Meet the MOBLO toys	Know about the MOBLO toys	Meet the unplugged programming toys	Know about the unplugged programming toys
	Sequence (1)	Understand the concept of "sequences"; know about the forward and backward Directional Blocks; be able to place the forward and backward Directional Blocks in sequence and develop a simple route/program for Kobe to defeat the monster.	Sequence (1)	Understand the concept of "sequences"; know about the forward and backward Directional Blocks; be able to place the forward and backward Directional Blocks in sequence and develop a simple route/program for Qiqi's tour.

Session 2	Sequence (2) and Decomposition (1)	1. Consolidate the concept of “sequences”; know about the left and right Directional Blocks; be able to place the left and right Directional Blocks in sequence and develop a simple route/program for Kobe to defeat the monster. 2. Observe the start and end points and be able to break down a route into several single steps.	Sequence (2) and Decomposition (1)	1. Consolidate the concept of “sequences”; know about the left and right Directional Blocks; be able to place the left and right Directional Blocks in sequence and develop a simple route/program for Qiqi’s tour. 2. Observe the start and end points and be able to break down a route into several single steps.
Session 3	Sequence (3) and Debugging (1)	1. Master the concept of “sequences” and be able to use the Directional Blocks (forward, backward, left and right) to develop a route/program for Kobe to defeat the monster.	Sequence (3) and Debugging (1)	1. Master the concept of “sequences” and be able to use the Directional Blocks (forward, backward, left and right) to develop a route/program for Qiqi’s tour.

		2. When an error occurs in the program, be able to check the sequence of Directional Blocks, find the wrong part and correct the error.		2. When an error occurs in the program, be able to check the sequence of Directional Blocks, find the wrong part and correct the error.
Session 4	Conditional (1) and Representation (1)	1. Understand the concept of conditionals; know about the Action Blocks and different tools and the need to use them when encountering special events; be able to use the Directional Blocks, Action Blocks and different tools to develop a route/program for Kobe to defeat the Monster. 2. Observe the symbols in the record column and understand the concept of representation; be able to use the symbols to represent the route Kobe takes.	Conditional (1) and Representation (1)	1. Understand the concept of conditionals; know about the Conditional Instruction Card and Tool Blocks and the need to use them when encountering special events; be able to use the Directional Blocks, Conditional Instruction Card and Tool Blocks to develop a route/program for Qiqi's tool. 2. Observe the symbols in the programming area and understand the concept of representation; be able to use

				the symbols to represent the route Qiqi takes.
Session 5	Conditional (2) and Decomposition (2)	1. Consolidate the concept of conditionals; be able to use the Directional Blocks, Action Blocks and different tools to develop a route/program for Kobe to defeat the Monster. 2. Be able to break down a problem into smaller easily solved parts.	Conditional (2) and Decomposition (2)	1. Consolidate the concept of conditionals; be able to use the Directional Blocks, Conditional Instruction Card and Tool Blocks to develop a route/program for Qiqi's tool. 2. Be able to break down a problem into smaller easily solved parts.
Session 6	Loops (1) and Representation (2)	1. Understand the concept of loops; be able to identify the repeating part and the number of repetitions in a route. 2. Observe the symbols in the record column and understand the concept of representation; be able to use the symbols to represent the route Kobe takes.	Loops (1) and Representation (2)	1. Understand the concept of loops; be able to identify the repeating part and the number of repetitions in a route. 2. Observe the symbols in the record column and understand the concept of

				representation; be able to use the symbols to represent the route Qiqi takes.
Session 7	Loops (2) and Debugging (2)	1. Consolidate the concept of loops; know about NFC Blocks and Number Cards; be able to identify the repeating part and the number of repetitions in a simple route and use NFC Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.	Loops (2) and Debugging (2)	1. Consolidate the concept of loops; know about Loops Blocks; be able to identify the repeating part and the number of repetitions in a simple route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 8	Loops (3) and Debugging (3)	1. Further consolidate the concept of loops; be able to identify the repeating part and the number of repetitions in a complex route and use NFC Blocks and Number Cards to input loops commands.	Loops (3) and Debugging (3)	1. Consolidate the concept of loops; be able to identify the repeating part and the number of repetitions in a complex route and use Loops Blocks to input loops commands.

		2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.		2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 9	Loops (4) and Debugging (4)	1. Master the concept of loops; be able to identify the repeating part and the number of repetitions in a more complex route and use NFC Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.	Loops (4) and Debugging (4)	1. Master the concept of loops; be able to identify the repeating part and the number of repetitions in a more complex route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 10	Loops (5) and Debugging (5)	1. Master the concept of loops; be able to quickly identify the repeating part and the number of repetitions in a more complex route and use NFC Blocks and Number Cards to input loops commands.	Loops (5) and Debugging (5)	1. Master the concept of loops; be able to quickly identify the repeating part and the number of repetitions in a more complex

		2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.		route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 11	The use of sequences, conditionals and loops (1) and algorithms (1)	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number Cards to develop a route/program for Kobe to defeat the Monster. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.	The use of sequences, conditionals and loops (2) and algorithms (2)	1. Be able to use Directional Blocks, Conditional Instruction Card, and Loops Blocks to develop a route/program for QiQi's tool. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.
Session 12	The use of sequences, conditionals	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number Cards to develop a route/program for Kobe to defeat the Monster.	The use of sequences, conditionals and	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number

and loops (2) and algorithms (2)	2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.	loops (2) and and algorithms (2)	Cards to develop a route/program for Kobe to defeat the Monster. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.

Note: Each activity takes about 40 minutes.

C) Potential benefits (including compensation for participation)

- Your child will receive free and effective computational thinking education in the classroom.

- Your child may benefit from the computational thinking activities with the improvement of computational thinking, coding skills, self-regulation, and more positive outcomes.

The potential risks of the research

- The study will present no more than minimal risk to the participants.

- The Research Assistant will be well-trained to provide comfortable experiences for your child in the assessments.

- Your child's involvement in the project is entirely voluntary. Both you and your child possess the autonomy to withdraw from the study at any point without encountering any adverse repercussions. All data pertaining to your child will be treated with utmost confidentiality and will be identified solely through unique codes known exclusively to the researcher.

How results will be potentially disseminated

- This project will help the participating children learn and develop early coding skills and computational thinking.

- Research results will be disseminated through thesis and journal article.

If you would like to obtain more information about this study, please contact ZENG Yue by email at [redacted]; Dr. Weipeng Yang by email at wyang@eduhk.hk.

If you or your child have/ has any concerns about the conduct of this research study, please do not hesitate to contact the Human Research Ethics Committee by email at hrec@eduhk.hk or by mail to Research and Development Office, The Education University of Hong Kong.

Thank you for your interest in participating in this study.

ZENG, Yue

November 1, 2022

香港教育大學

幼兒教育學系

參與研究同意書（家長）

插電式和非插電式編程課程對 5-6 歲兒童的計算思維、自我調節、心流體驗和自我效能感的影響

茲同意_____（兒童姓名）參加由楊偉鵬博士和 Alfredo Bautista 博士負責監督,曾越負責執行的研究計畫。他/她們是香港教育大學幼稚教育系的教員/学生。

本人理解此研究所獲得的資料可用於未來的研究和學術發表。然而本人有權保護敝子女的隱私,其個人資料將不能洩漏。

研究者已將所附資料的有關步驟向本人作了充分的解釋。本人理解可能會出現的風險。本人是自願讓敝子女參與這項研究。

本人理解本人及敝子女皆有權在研究過程中提出問題,並在任何時候決定退出研究, 更不會因此而對研究工作產生的影響負有任何責任。

參加者姓名 _____

父母姓名或監護人姓名: _____

父母或監護人簽名: _____

日期: _____

有關資料

插電式和非插電式編程課程對 5-6 歲兒童的計算思維、自我調節、心流體驗和自我效能感的影響

誠邀貴子女參加楊偉鵬博士和 Alfredo Bautista 博士負責監督,曾越負責執行的研究計畫。他/她們是香港教育大學幼稚教育系的教員/学生。

研究計畫簡介

- 我們將為幼兒設計一系列插電和不插電編程活動，幫助幼兒學習順序、迴圈、條件、問題分解、調試等編程概念和技能。不插電的編程課程指的是沒有數字設備的編程教學，通常涉及紙和筆、卡片、貼紙書以及身體動作，而插電的編程課程指的是使用數字設備的編程教學。在插電的編程活動中，我們將使用 MBOLO 編程教具進行編程教學；在不插電的編程活動中，我們將使用無螢幕編程材料進行編程教學；
- 我們將為教師提供有關什麼是編程、如何開展編程活動的培訓；
- 我們將評估插電和不插電的編程課程對幼兒計算思維、自我調節能力、心流體驗、編程自我效能感的影響；
- 實驗班的教師將負責開展編程課程，我們將對兩個實驗班進行的

編程活動（大約兩個月）進行錄影；

- 在所有的編程活動結束後，我們將對實驗班的兩位老師和個別幼兒進行訪談，所有的訪談將被錄音。

研究方法

A)　工作及步驟

- 每個孩子的計算思維將在編程課程（干預）前後被評估（前測和後測）。計算思維評估通常需要 12 分鐘，由研究人員在幼稚園的一個安靜的房間裏進行。

- 每個孩子的自我調節能力將在編程課程（干預）前後被評估（前測和後測）。自我調節評估通常需要 15 分鐘，由研究人員在幼稚園的一個安靜的房間裏對幼兒進行。

- 每個孩子的流動體驗將在編程課程（干預）後被評估。 評估將需要大約兩分鐘，由研究人員在幼稚園的一個安靜房間裏進行。

- 每個孩子的編程自我效能感將在編程專案（干預）後被評估。編程自我效能評估將由班主任老師完成。

- 焦點小組訪談將分別與每個實驗組的 10 名兒童進行。訪談將被錄影，並將持續約一個小時。

- 老師將進行編程活動（每個編程活動的細節見表 1），研究人員將對所有的編程活動進行錄影（12 節，每節 50 分鐘）。

表 1

編程課程

	插电式编程活动	活动目标	不插电编程活动	活动目标
Session 1	《認識 MOBLO 玩具》 《順序 1》	瞭解“順序”的概念；認識方向積木；能夠有順序地擺放方向積木，編寫科比打敗怪獸神的路線/程式。	《認識 unplugged programming 玩具》 《順序 1》	瞭解“順序”的概念；認識方向積木；能夠有順序地擺放方向積木，編寫奇奇旅遊的路線/程式。
Session 2	《順序 2》 《問題分解 1》	鞏固“順序”的概念；能夠較為熟練地運用方向積木編寫路線/程式。	《順序 2》 《問題分解 1》	鞏固“順序”的概念；能夠較為熟練地運用方向積木編寫路線/程式。

		在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。		在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。
Session 3	《順序 3》 《調試 1》	掌握“順序”的概念；能夠熟練運用方向積木編寫更長的路線/程式。 當編寫的路線/程式出現錯誤時，能夠檢查方向積木或記錄欄中的程式，找出錯誤的部分並糾正錯誤。	《順序 3》 《調試 1》	掌握“順序”的概念；能夠熟練運用方向積木編寫更長的路線/程式。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。

Session 4	《條件 1》 《表徵 1》	瞭解“條件”的概念；認識“動作積木”，瞭解在遇到特殊事件時需要使用動作積木；初步學習觀察路線，判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編寫科比打敗怪獸神的路線/程式。 觀察記錄欄中的符號，理解表徵的概念：使用抽象的符號來表示具體的指令。	《條件 1》 《表徵 1》	瞭解“條件”的概念；認識“條件指令卡”和工具積木，瞭解在遇到特殊事件時需要使用“條件指令卡”和工具積木；初步學習判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 觀察編程區的指令，理解表徵的概念：使用抽象的符號來表示具體的指令。

Session 5	《條件 2》 《問題分解 2》	鞏固“條件”的概念；能夠較為熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編寫科比打敗怪獸神的路線/程式。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。	《條件 2》 《問題分解 2》	鞏固“條件”的概念；能夠較為熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。
Session 6	《條件 3》 《調試 2》	進一步鞏固“條件”的概念；能夠熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編	《條件 3》 《調試 2》	進一步鞏固“條件”的概念；能夠熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條

		寫科比打敗怪獸神的路線/程式。 當編寫的路線/程式出現錯誤時，能夠檢查動作積木和方向積木或記錄欄中的程式，找出錯誤的部分並糾正錯誤。		件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。
Session 7	《反復 1》 《表徵 2》	瞭解“反復”的概念；認識 NFC 積木和數字卡片；能夠找出簡單的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。	《反復 1》 《表徵 2》	瞭解“反復”的概念；認識反復積木；能夠找出簡單的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。

		觀察記錄欄中的符號，理解表徵的概念：使用抽象的符號來表示具體的指令。		觀察編程區的指令，理解表徵的概念：使用抽象的符號來表示具體的指令。
Session 8	《反復 2》 《問題分解 3》	鞏固“反復”的概念；能夠找出比較簡單的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。	《反復 2》 《問題分解 3》	鞏固“反復”的概念；能夠找出比較簡單的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。

Session 9	《反復 3》 《調試 3》	進一步鞏固“反復”的概念；能夠找出比較複雜的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。 當編寫的路線/程式出現錯誤時，能夠檢查方向積木和數字卡片或記錄欄中的程式，找出錯誤的部分並糾正錯誤。	《反復 3》 《調試 3》	進一步鞏固“反復”的概念；能夠找出比較複雜的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。
Session 10	《反復 4》	掌握“反復”的概念；能夠比較快速地找出複雜的反復路線中的反復部分和反復次數，並利	《反復 4》	掌握“反復”的概念；能夠比較快速地找出複雜的反復路線中的反復部分和

		用 NFC 積木和數字卡片，輸入迴圈命令語。		反復次數，並利用反復積木輸入迴圈命令語。
Session 11	《順序、條件和反復的綜合運用 1》 《演算法 1》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。 理解演算法的概念，並能運用順序、條件和反復等積木進行較為簡單的演算法設計。	《順序、條件和反復的綜合運用 1》 《演算法 1》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。 理解演算法的概念，並能運用順序、條件和反復等積木進行較為簡單的演算法設計。
Session 12	《順序、條件和反復的綜合運用 2》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。	《順序、條件和反復的綜合運用 2》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。

《演算法 2》	理解演算法的概念，並能運用順序、條件和反復等積木進行較為複雜的演算法設計。	《演算法 2》	理解演算法的概念，並能運用順序、條件和反復等積木進行較為複雜的演算法設計。

注：每個活動約需 50 分鐘。

任何利益 (包括對參與者的補償)

- 貴子女將在幼稚園接受免費有效的編程教育。

- 貴子女可能會從計算思維活動中受益，提高計算思維、編程技能、自我調節能力和更多其他積極的結果。

參與期間有可能面對的風險及不適

- 該研究對參與者的風險極低。

- 研究助理將接受良好培訓，為您的孩子提供舒適的評估體驗。

- 閣下及貴子女的參與純屬自願性質。閣下及貴子女享有充分的權利在任何時候決定退出這項研究,更不會因此引致任何不良後果。凡有關貴子女的資料將會保密,一切資料的編碼只有研究人員得悉。

將如何發佈研究結果

- 該專案將幫助參與的兒童學習和發展早期的編程技能和計算思維。

- 研究成果將通過畢業論文、期刊論文傳播。

如閣下想獲得更多有關這項研究的資料,請電郵與曾越(s1142522@s.eduhk.hk)聯絡。

如閣下或 貴子女對這項研究的研究倫理有任何意見,可隨時與香港教育大學

人類實驗對象研究倫理委員會聯絡(電郵: hrec@eduhk.hk ; 地址:香港教育大學研究與發展事務處)。

謝謝閣下有興趣參與這項研究。

曾越

2022 年 11 月 1 日

Consent Form and Information Sheet for PARTICIPANTS

THE EDUCATION UNIVERSITY OF HONG KONG

Department of Early Childhood Education

CONSENT TO PARTICIPATE IN RESEARCH

Effects of Plugged and Unplugged Programming Curricula on Computational Thinking, Self-Regulation, Flow Experience and Self-Efficacy of Children Aged 5-6

I _____ , hereby provide my informed consent to participate in a research project under the supervision of Dr. Weipeng Yang and Dr. Alfredo Bautista and conducted by Yue Zeng, who are affiliated with the Department of Early Childhood Education at The Education University of Hong Kong.

I am aware that the data collected from this research may be utilized in future studies and potentially published. However, my privacy will be safeguarded, ensuring that my personal information remains confidential.

I have been thoroughly briefed on the procedure outlined in the attached information sheet. I understand the potential benefits and risks associated with my participation. I confirm that my involvement in this project is voluntary.

I acknowledge my right to raise any concerns or queries regarding any aspect of the research procedure and retain the freedom to withdraw my participation at any time without encountering any adverse consequences.

Name of participant _____

Signature of participant _____

Date _____

INFORMATION SHEET

Effects of Plugged and Unplugged Programming Curricula on Computational Thinking, Self-Regulation, Flow Experience and Self-Efficacy of Children Aged 5-6

You are invited to participate in a project supervised by Dr. Weipeng Yang and Dr. Alfredo Bautista and conducted by Yue Zeng, who are staff / students of the department of Early Childhood Education in The Education University of Hong Kong.

The introduction of the research

- I will design a series of plug-in and unplugged programming activities for children to help them learn sequencing, loops, conditionals, decomposition, debugging, and other programming concepts and skills. Unplugged programming curriculum refers to teaching programming without digital devices and often involves paper and pencil, cards, sticker books, as well as body movements, while plugged programming curriculum refers to teaching programming with the use of digital devices. I will use MBOLO in the plugged programming group and use unplugged materials in the unplugged programming group.

- I will provide training to teachers on what computational thinking is and how to develop it in young children; and

- I will evaluate the impact of plugged and unplugged programming courses on children's computational thinking, self-regulation skills, flow experiences, and programming self-efficacy.

- I will videotape the programming activities (about two months) carried out by the two experimental classes.

- After all programming activities, interviews will be conducted with two teachers and several children from the experimental classes, and all interviews will be recorded.

The methodology of the research

A) Procedure of the research

- You will receive 2 hours of training in the computational thinking program during non-work time slots one week before the programming curriculum.

- The computational thinking course (intervention) will last about two months. Each session will last approximately 40 minutes, twice a week (the details of each programming activity see Table 1). You will be responsible for teaching the computational thinking activities.

- At the end of the programming curriculum, you will be interviewed for approximately 60 minutes. The interview will be used to learn about your attitudes towards teaching computational thinking to young children. The interviews will take place in a quiet room at

your school.

- Each child's computational thinking will be assessed (pre-test and post-test) before and after the programming curriculum (intervention). The computational thinking assessment typically takes 12 minutes to administer to young children and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's self-regulation skills will be assessed (pre-test and post-test) before and after the programming curriculum (intervention). The self-regulation assessment usually takes 15 minutes to administer to young children and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's flow experience will be assessed after the programming program (intervention). The assessment will take about two minutes and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's programming self-efficacy will be assessed after the programming program (intervention). The programming self-efficacy assessment will be completed by you. It will take about one hour.

- Focus group interviews will be conducted with respectively ten children from each experimental group. The interviews will be videotaped and will last about one hour.

Table 1

Programming Courses

Session	**Plugged activities**		**Unplugged activities**	
	Activity	**Objective**	**Activity**	**Objective**
Session 1	Meet the MOBLO toys	Know about the MOBLO toys	Meet the unplugged programming toys	Know about the unplugged programming toys
	Sequence (1)	Understand the concept of "sequences"; know about the forward and backward Directional Blocks; be able to place the forward and backward Directional Blocks in sequence and develop a simple route/program for Kobe to defeat the monster.	Sequence (1)	Understand the concept of "sequences"; know about the forward and backward Directional Blocks; be able to place the forward and backward Directional Blocks in sequence and develop a simple route/program for Qiqi's tour.

Session 2	Sequence (2) and Decomposition (1)	1. Consolidate the concept of “sequences”; know about the left and right Directional Blocks; be able to place the left and right Directional Blocks in sequence and develop a simple route/program for Kobe to defeat the monster. 2. Observe the start and end points and be able to break down a route into several single steps.	Sequence (2) and Decomposition (1)	1. Consolidate the concept of “sequences”; know about the left and right Directional Blocks; be able to place the left and right Directional Blocks in sequence and develop a simple route/program for Qiqi’s tour. 2. Observe the start and end points and be able to break down a route into several single steps.
Session 3	Sequence (3) and Debugging (1)	1. Master the concept of “sequences” and be able to use the Directional Blocks (forward, backward, left and right) to develop a route/program for Kobe to defeat the monster. 2. When an error occurs in the program, be able to check the	Sequence (3) and Debugging (1)	1. Master the concept of “sequences” and be able to use the Directional Blocks (forward, backward, left and right) to develop a route/program for Qiqi’s tour.

		sequence of Directional Blocks, find the wrong part and correct the error.		2. When an error occurs in the program, be able to check the sequence of Directional Blocks, find the wrong part and correct the error.
Session 4	Conditional (1) and Representation (1)	1. Understand the concept of conditionals; know about the Action Blocks and different tools and the need to use them when encountering special events; be able to use the Directional Blocks, Action Blocks and different tools to develop a route/program for Kobe to defeat the Monster. 2. Observe the symbols in the record column and understand the concept of representation; be able to use the symbols to represent the route Kobe takes.	Conditional (1) and Representation (1)	1. Understand the concept of conditionals; know about the Conditional Instruction Card and Tool Blocks and the need to use them when encountering special events; be able to use the Directional Blocks, Conditional Instruction Card and Tool Blocks to develop a route/program for Qiqi's tool. 2. Observe the symbols in the programming area and understand the concept of representation; be able to use

				the symbols to represent the route Qiqi takes.
Session 5	Conditional (2) and Decomposition (2)	1. Consolidate the concept of conditionals; be able to use the Directional Blocks, Action Blocks and different tools to develop a route/program for Kobe to defeat the Monster. 2. Be able to break down a problem into smaller easily solved parts.	Conditional (2) and Decomposition (2)	1. Consolidate the concept of conditionals; be able to use the Directional Blocks, Conditional Instruction Card and Tool Blocks to develop a route/program for Qiqi's tool. 2. Be able to break down a problem into smaller easily solved parts.
Session 6	Loops (1) and Representation (2)	1. Understand the concept of loops; be able to identify the repeating part and the number of repetitions in a route. 2. Observe the symbols in the record column and understand the concept of representation; be able to use the symbols to represent the route Kobe takes.	Loops (1) and Representation (2)	1. Understand the concept of loops; be able to identify the repeating part and the number of repetitions in a route. 2. Observe the symbols in the record column and understand the concept of

				representation; be able to use the symbols to represent the route Qiqi takes.
Session 7	Loops (2) and Debugging (2)	1. Consolidate the concept of loops; know about NFC Blocks and Number Cards; be able to identify the repeating part and the number of repetitions in a simple route and use NFC Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.	Loops (2) and Debugging (2)	1. Consolidate the concept of loops; know about Loops Blocks; be able to identify the repeating part and the number of repetitions in a simple route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 8	Loops (3) and Debugging (3)	1. Further consolidate the concept of loops; be able to identify the repeating part and the number of repetitions in a complex route and use NFC Blocks and Number Cards to input loops commands.	Loops (3) and Debugging (3)	1. Consolidate the concept of loops; be able to identify the repeating part and the number of repetitions in a complex route and use Loops Blocks to input loops commands.

		2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.		2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 9	Loops (4) and Debugging (4)	1. Master the concept of loops; be able to identify the repeating part and the number of repetitions in a more complex route and use NFC Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.	Loops (4) and Debugging (4)	1. Master the concept of loops; be able to identify the repeating part and the number of repetitions in a more complex route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 10	Loops (5) and Debugging (5)	1. Master the concept of loops; be able to quickly identify the repeating part and the number of repetitions in a more complex route and use NFC Blocks and Number Cards to input loops commands.	Loops (5) and Debugging (5)	1. Master the concept of loops; be able to quickly identify the repeating part and the number of repetitions in a more complex

		2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.		route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 11	The use of sequences, conditionals and loops (1) and algorithms (1)	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number Cards to develop a route/program for Kobe to defeat the Monster. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.	The use of sequences, conditionals and loops (2) and algorithms (2)	1. Be able to use Directional Blocks, Conditional Instruction Card, and Loops Blocks to develop a route/program for QiQi's tool. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.
Session 12	The use of sequences, conditionals	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number Cards to develop a route/program for Kobe to defeat the Monster.	The use of sequences, conditionals and	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number

and loops (2) and algorithms (2)	2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.	loops (2) and and algorithms (2)	Cards to develop a route/program for Kobe to defeat the Monster. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.

Note: Each activity takes about 40 minutes.

B) Potential benefits (including compensation for participation)

- You will receive free training on computational thinking education in early childhood.

- The kindergarten will have the opportunity to incorporate the most advanced computational thinking education into the school-based curriculum by working with the Wenzhou University research team.

- The kindergarten and teachers will learn how to look for appropriate **curriculum materials and resources** for delivering computational thinking education.

- You will receive a gift worth 300 RMB.

The potential risks associated with the research include:

-The study poses minimal risk to participants.

-Your participation is voluntary, and you have the right to withdraw without facing any negative consequences.

-All information collected will be kept confidential and identifiable only by unique codes known solely to the researcher.

How results will be potentially disseminated

-The project will offer teacher training and curriculum resources to participating kindergartens.

-Research findings will be shared through a thesis and journal articles.

For further information about this study, please contact ZENG Yue via email at [redacted] or Dr. Weipeng Yang at wyang@eduhk.hk.

If you have any concerns regarding the ethical conduct of this research study, please don't hesitate to contact the Human Research Ethics Committee via email at hrec@eduhk.hk or by mail at the Research and Development Office, The Education University of Hong Kong.

We appreciate your interest in participating in this study.

ZENG, Yue

November 1, 2022

香港教育大學

幼兒教育學系

參與研究同意書（教師）

插電式和非插電式編程課程對 5-6 歲兒童的計算思維、自我調節、心流體驗和自我效能感的影響

本人_____同意參加由楊偉鵬博士和 Alfredo Bautista 博士負責監督,曾越負責執行的研究計畫。他/她們是香港教育大學幼稚教育系的教員/学生。

本人理解此研究所獲得的資料可用於未來的研究和學術發表。然而本人有權保護自己的隱私, 本人的個人資料將不能洩漏。

研究者已將所附資料的有關步驟向本人作了充分的解釋。本人理解可能會出現的風險。本人是自願參與這項研究。

本人理解我有權在研究過程中提出問題,並在任何時候決定退出研究,更不會因此而對研究工作產生的影響負有任何責任。

參加者姓名: _____

參加者簽名: _____

日期: _____

有關資料

插電式和非插電式編程課程對 5-6 歲兒童的計算思維、自我調節、心流體驗和自我效能感的影響

誠邀閣下參加楊偉鵬博士和 Alfredo Bautista 博士負責監督,曾越負責執行的研究計畫。他/她們是香港教育大學幼稚教育系的教員/学生。

研究計畫簡介

- 我們將為幼兒設計一系列插電和不插電編程活動，幫助幼兒學習順序、迴圈、條件、問題分解、調試等編程概念和技能。不插電的編程課程指的是沒有數字設備的編程教學，通常涉及紙和筆、卡片、貼紙書以及身體動作，而插電的編程課程指的是使用數字設備的編程教學。在插電的編程活動中，我們將使用 MBOLO 編程教具進行編程教學；在不插電的編程活動中，我們將使用無螢幕編程材料進行編程教學；
- 我們將為教師提供有關什麼是編程、如何開展編程活動的培訓；
- 我們將評估插電和不插電的編程課程對幼兒計算思維、自我調節能力、心流體驗、編程自我效能感的影響；
- 我們將對兩個實驗班進行的編程活動（大約兩個月）進行錄影；

- 在所有的編程活動結束後，我們將對實驗班的兩位老師和個別幼兒進行訪談，所有的訪談將被錄音。

研究方法

工作及步驟

- 在編程課程開始前一周，您將在非工作的時間段接受 2 小時的編程課程培訓。

- 編程活動（干預）將持續兩個月。每個星期兩次，每次大約 50 分鐘。您將承擔編程活動的教學工作（編程課程詳見表 1）。

- 在編程課程結束之後，您將接受大約 60 分鐘的訪談。訪談將用於瞭解您對幼兒編程教育的態度。訪談都將在貴校安靜的房間內進行。訪談將被錄音。

- 每個孩子的計算思維將在編程課程（干預）前後被評估（前測和後測）。計算思維評估通常需要 12 分鐘，由研究人員在幼稚園的一個安靜的房間裏進行。

- 每個孩子的自我調節能力將在編程課程（干預）前後被評估（前測和後測）。自我調節評估通常需要 15 分鐘，由研究人員在幼稚園的一個安靜的房間裏對幼兒進行。

- 每個孩子的流動體驗將在編程課程（干預）後被評估。 評估將需要大約兩分鐘，由研究人員在幼稚園的一個安靜房間裏進行。

- 每個孩子的編程自我效能感將在編程專案（干預）後被評估。編程自我效能評估將由你來完成。這將需要大約一個小時。

- 焦點小組訪談將分別與每個實驗組的 10 名兒童進行。訪談將被錄影，並將持續約一個小時。

表 1

編程課程

	插电式编程活动	活动目标	不插电编程活动	活动目标
Session 1	《認識 MOBLO 玩具》 《順序 1》	瞭解“順序”的概念；認識方向積木；能夠有順序地擺放方向積木，編寫科比打敗怪獸神的路線/程式。	《認識 unplugged programming 玩具》 《順序 1》	瞭解“順序”的概念；認識方向積木；能夠有順序地擺放方向積木，編寫奇奇旅遊的路線/程式。
Session 2	《順序 2》 《問題分解 1》	鞏固“順序”的概念；能夠較為熟練地運用方向積木編寫路線/程式。	《順序 2》 《問題分解 1》	鞏固“順序”的概念；能夠較為熟練地運用方向積木編寫路線/程式。

		在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。		在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。
Session 3	《順序 3》 《調試 1》	掌握“順序”的概念；能夠熟練運用方向積木編寫更長的路線/程式。 當編寫的路線/程式出現錯誤時，能夠檢查方向積木或記錄欄中的程式，找出錯誤的部分並糾正錯誤。	《順序 3》 《調試 1》	掌握“順序”的概念；能夠熟練運用方向積木編寫更長的路線/程式。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。

Session 4	《條件 1》 《表徵 1》	瞭解“條件”的概念；認識“動作積木”，瞭解在遇到特殊事件時需要使用動作積木；初步學習觀察路線，判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編寫科比打敗怪獸神的路線/程式。 觀察記錄欄中的符號，理解表徵的概念：使用抽象的符號來表示具體的指令。	《條件 1》 《表徵 1》	瞭解“條件”的概念；認識“條件指令卡”和工具積木，瞭解在遇到特殊事件時需要使用“條件指令卡”和工具積木；初步學習判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 觀察編程區的指令，理解表徵的概念：使用抽象的符號來表示具體的指令。

Session 5	《條件 2》 《問題分解 2》	鞏固“條件”的概念；能夠較為熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編寫科比打敗怪獸神的路線/程式。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。	《條件 2》 《問題分解 2》	鞏固“條件”的概念；能夠較為熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。
Session 6	《條件 3》 《調試 2》	進一步鞏固“條件”的概念；能夠熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編	《條件 3》 《調試 2》	進一步鞏固“條件”的概念；能夠熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條

		寫科比打敗怪獸神的路線/程式。 當編寫的路線/程式出現錯誤時，能夠檢查動作積木和方向積木或記錄欄中的程式，找出錯誤的部分並糾正錯誤。		件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。
Session 7	《反復 1》 《表徵 2》	瞭解“反復”的概念；認識 NFC 積木和數字卡片；能夠找出簡單的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。	《反復 1》 《表徵 2》	瞭解“反復”的概念；認識反復積木；能夠找出簡單的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。

		觀察記錄欄中的符號，理解表徵的概念：使用抽象的符號來表示具體的指令。		觀察編程區的指令，理解表徵的概念：使用抽象的符號來表示具體的指令。
Session 8	《反復 2》 《問題分解 3》	鞏固“反復”的概念；能夠找出比較簡單的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。	《反復 2》 《問題分解 3》	鞏固“反復”的概念；能夠找出比較簡單的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。

Session 9	《反復 3》 《調試 3》	進一步鞏固“反復”的概念；能夠找出比較複雜的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。 當編寫的路線/程式出現錯誤時，能夠檢查方向積木和數字卡片或記錄欄中的程式，找出錯誤的部分並糾正錯誤。	《反復 3》 《調試 3》	進一步鞏固“反復”的概念；能夠找出比較複雜的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。
Session 10	《反復 4》	掌握“反復”的概念；能夠比較快速地找出複雜的反復路線中的反復部分和反復次數，並利	《反復 4》	掌握“反復”的概念；能夠比較快速地找出複雜的反復路線中的反復部分和

		用 NFC 積木和數字卡片，輸入迴圈命令語。		反復次數，並利用反復積木輸入迴圈命令語。
Session 11	《順序、條件和反復的綜合運用 1》 《演算法 1》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。 理解演算法的概念，並能運用順序、條件和反復等積木進行較為簡單的演算法設計。	《順序、條件和反復的綜合運用 1》 《演算法 1》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。 理解演算法的概念，並能運用順序、條件和反復等積木進行較為簡單的演算法設計。
Session 12	《順序、條件和反復的綜合運用 2》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。	《順序、條件和反復的綜合運用 2》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。

《演算法 2》	理解演算法的概念，並能運用順序、條件和反復等積木進行較為複雜的演算法設計。	《演算法 2》	理解演算法的概念，並能運用順序、條件和反復等積木進行較為複雜的演算法設計。

注：每個活動約需 50 分鐘。

任何利益（包括對參與者的補償）

- 您將接受有關早期編程教育的免費培訓。

- 幼稚園將有機會與溫大研究團隊合作，將最先進的編程教育納入園本課程。

- 幼稚園和教師將學習如何尋找合適的課程材料和資源來設計編程活動。

- 您將收到價值 300 元的禮品一份。

參與期間可能面臨的風險與不適：

-這項研究對參與者的風險非常低。

-貴園學生/教師的參與完全是自願的。所有參與者在研究開始前或結束後都有權利選擇退出，並不會有任何不良後果。貴園學生/教師的相關資料將被保密，只有研究人員能夠讀取編碼後的資料。

研究結果的發佈方式：

-本項目將為參與的幼稚園提供有关编程教育的教師培訓和課程資源。

-研究結果將透過畢業論文和期刊論文來傳播。

如果您想獲得更多關於這項研究的資訊，請聯繫曾越（[redacted]）。

如果您對這項研究的研究倫理有任何意見，請隨時聯繫香港教育大學人類實驗對象研

究倫理委員會（電郵：hrec@eduhk.hk；地址：香港教育大學研究與發展事務處）。

謝謝您對參與這項研究的興趣。

曾越

2022 年 11 月 1 日

Consent Form and Information Sheet for SCHOOLS

THE EDUCATION UNIVERSITY OF HONG KONG

Department of Early Childhood Education

CONSENT TO PARTICIPATE IN RESEARCH

Effects of Plugged and Unplugged Programming Curricula on Computational Thinking, Self-Regulation, Flow Experience and Self-Efficacy of Children Aged 5-6

My school hereby consents to participate in a project supervised by Dr. Weipeng Yang and Dr. Alfredo Bautista and conducted by Yue Zeng, who are staff / students of the department of Early Childhood Education in The Education University of Hong Kong.

I understand that information obtained from this research may be used in future research and may be published. However, our right to privacy will be retained, i.e., the personal details of my students'/teachers' will not be revealed.

The procedure as set out in the **attached** information sheet has been fully explained. I understand the benefits and risks involved. My students'/teachers' participation in the project

are voluntary.

I acknowledge that we have the right to question any part of the procedure and can withdraw at any time without negative consequences.

Signature: _____

Name of Principal/Delegate*: _____ (Prof/Dr/Mr/Mrs/Ms/Miss*)

Post: _____

Name of School: _____

Date: _____

(please delete as appropriate)*

INFORMATION SHEET

Effects of Plugged and Unplugged Programming Curricula on Computational Thinking, Self-Regulation, Flow Experience and Self-Efficacy of Children Aged 5-6

Your kindergarten is invited to participate in a project supervised by Dr. Weipeng Yang and Dr. Alfredo Bautista and conducted by Yue Zeng, who are staff / students of the department of Early Childhood Education in The Education University of Hong Kong.

The introduction of the research

- I will design a series of plug-in and unplugged programming activities for children to help them learn sequencing, loops, conditionals, decomposition, debugging, and other programming concepts and skills. Unplugged programming curriculum refers to teaching programming without digital devices and often involves paper and pencil, cards, sticker books, as well as body movements, while plugged programming curriculum refers to teaching programming with the use of digital devices. I will use MBOLO in the plugged programming group and use unplugged materials in the unplugged programming group.
- I will provide training to teachers on what computational thinking is and how to develop it in

young children; and

- I will evaluate the impact of plugged and unplugged programming courses on children's computational thinking, self-regulation skills, flow experiences, and programming self-efficacy.

- I will videotape the programming activities (about two months) carried out by the two experimental classes.

- After all programming activities, interviews will be conducted with two teachers and several children from the experimental classes, and all interviews will be recorded.

The methodology of the research

A) Procedure of the research

- The teachers will receive 2 hours of training in the computational thinking program during non-work time slots one week before the computational thinking curriculum.

- The computational thinking course (intervention) will last for two months. Each session will last approximately 40 minutes, twice a week (the details of each programming activity see Table 1). The teachers of the experimental classes will be responsible for teaching the computational thinking activities.

- At the end of the computational thinking curriculum, the teachers will be interviewed for approximately 60 minutes. The interview will be used to learn about your attitudes towards teaching computational thinking to young children. The interviews will take place in a quiet

room at your school.

- Each child's computational thinking will be assessed (pre-test and post-test) before and after the programming curriculum (intervention). The computational thinking assessment typically takes 12 minutes to administer to young children and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's self-regulation skills will be assessed (pre-test and post-test) before and after the programming curriculum (intervention). The self-regulation assessment usually takes 15 minutes to administer to young children and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's flow experience will be assessed after the programming program (intervention). The assessment will take about two minutes and will be conducted by the researcher in a quiet room in the kindergarten.

- Each child's programming self-efficacy will be assessed after the programming program (intervention). The programming self-efficacy assessment will be completed by the teacher. It will take about one hour.

- Focus group interviews will be conducted with respectively ten children from each experimental group. The interviews will be videotaped and will last about one hour.

Table 1

Programming Courses

Session	**Plugged activities**		**Unplugged activities**	
	Activity	**Objective**	**Activity**	**Objective**
Session 1	Meet the MOBLO toys	Know about the MOBLO toys	Meet the unplugged programming toys	Know about the unplugged programming toys
	Sequence (1)	Understand the concept of "sequences"; know about the forward and backward Directional Blocks; be able to place the forward and backward Directional Blocks in sequence and develop a simple route/program for Kobe to defeat the monster.	Sequence (1)	Understand the concept of "sequences"; know about the forward and backward Directional Blocks; be able to place the forward and backward Directional Blocks in sequence and develop a simple route/program for Qiqi's tour.

Session 2	Sequence (2) and Decomposition (1)	1. Consolidate the concept of “sequences”; know about the left and right Directional Blocks; be able to place the left and right Directional Blocks in sequence and develop a simple route/program for Kobe to defeat the monster. 2. Observe the start and end points and be able to break down a route into several single steps.	Sequence (2) and Decomposition (1)	1. Consolidate the concept of “sequences”; know about the left and right Directional Blocks; be able to place the left and right Directional Blocks in sequence and develop a simple route/program for Qiqi’s tour. 2. Observe the start and end points and be able to break down a route into several single steps.
Session 3	Sequence (3) and Debugging (1)	1. Master the concept of “sequences” and be able to use the Directional Blocks (forward, backward, left and right) to develop a route/program for Kobe to defeat the monster.	Sequence (3) and Debugging (1)	1. Master the concept of “sequences” and be able to use the Directional Blocks (forward, backward, left and right) to develop a route/program for Qiqi’s tour.

		2. When an error occurs in the program, be able to check the sequence of Directional Blocks, find the wrong part and correct the error.		2. When an error occurs in the program, be able to check the sequence of Directional Blocks, find the wrong part and correct the error.
Session 4	Conditional (1) and Representation (1)	1. Understand the concept of conditionals; know about the Action Blocks and different tools and the need to use them when encountering special events; be able to use the Directional Blocks, Action Blocks and different tools to develop a route/program for Kobe to defeat the Monster. 2. Observe the symbols in the record column and understand the concept of representation; be able to use the symbols to represent the route Kobe takes.	Conditional (1) and Representation (1)	1. Understand the concept of conditionals; know about the Conditional Instruction Card and Tool Blocks and the need to use them when encountering special events; be able to use the Directional Blocks, Conditional Instruction Card and Tool Blocks to develop a route/program for Qiqi's tool. 2. Observe the symbols in the programming area and understand the concept of representation; be able to use

				the symbols to represent the route Qiqi takes.
Session 5	Conditional (2) and Decomposition (2)	1. Consolidate the concept of conditionals; be able to use the Directional Blocks, Action Blocks and different tools to develop a route/program for Kobe to defeat the Monster. 2. Be able to break down a problem into smaller easily solved parts.	Conditional (2) and Decomposition (2)	1. Consolidate the concept of conditionals; be able to use the Directional Blocks, Conditional Instruction Card and Tool Blocks to develop a route/program for Qiqi's tool. 2. Be able to break down a problem into smaller easily solved parts.
Session 6	Loops (1) and Representation (2)	1. Understand the concept of loops; be able to identify the repeating part and the number of repetitions in a route. 2. Observe the symbols in the record column and understand the concept of representation; be able	Loops (1) and Representation (2)	1. Understand the concept of loops; be able to identify the repeating part and the number of repetitions in a route. 2. Observe the symbols in the record column and understand the concept of

		to use the symbols to represent the route Kobe takes.		representation; be able to use the symbols to represent the route Qiqi takes.
Session 7	Loops (2) and Debugging (2)	1. Consolidate the concept of loops; know about NFC Blocks and Number Cards; be able to identify the repeating part and the number of repetitions in a simple route and use NFC Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.	Loops (2) and Debugging (2)	1. Consolidate the concept of loops; know about Loops Blocks; be able to identify the repeating part and the number of repetitions in a simple route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 8	Loops (3) and Debugging (3)	1. Further consolidate the concept of loops; be able to identify the repeating part and the number of repetitions in a complex route and use NFC	Loops (3) and Debugging (3)	1. Consolidate the concept of loops; be able to identify the repeating part and the number of repetitions in a complex route

		Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.		and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 9	Loops (4) and Debugging (4)	1. Master the concept of loops; be able to identify the repeating part and the number of repetitions in a more complex route and use NFC Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.	Loops (4) and Debugging (4)	1. Master the concept of loops; be able to identify the repeating part and the number of repetitions in a more complex route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 10	Loops (5) and Debugging (5)	1. Master the concept of loops; be able to quickly identify the repeating part and the number of	Loops (5) and Debugging (5)	1. Master the concept of loops; be able to quickly identify the repeating part and

		repetitions in a more complex route and use NFC Blocks and Number Cards to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.		the number of repetitions in a more complex route and use Loops Blocks to input loops commands. 2. When an error occurs in the program, be able to check program, find the wrong part and correct the error.
Session 11	The use of sequences, conditionals and loops (1) and algorithms (1)	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number Cards to develop a route/program for Kobe to defeat the Monster. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.	The use of sequences, conditionals and loops (2) and algorithms (2)	1. Be able to use Directional Blocks, Conditional Instruction Card, and Loops Blocks to develop a route/program for QiQi's tool. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals and loops.

Session 12	The use of sequences, conditionals, and loops (2) and algorithms (2)	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number Cards to develop a route/program for Kobe to defeat the Monster. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals, and loops.	The use of sequences, conditionals, and loops (2) and algorithms (2)	1. Be able to use Directional Blocks, Action Blocks, and NFC and Number Cards to develop a route/program for Kobe to defeat the Monster. 2. Understand the concept of algorithms and be able to design simple algorithms using sequences, conditionals, and loops.

Note: Each activity takes about 40 minutes.

C) Potential benefits (including compensation for participation)

- Teachers will receive free training on computational thinking education in early childhood.

- The kindergarten will have the opportunity to incorporate the most advanced computational thinking education into the school-based curriculum by working with the Wenzhou University research team.

- The kindergarten and teachers will learn how to look for appropriate **curriculum materials and resources** for delivering computational thinking education.

- The teacher in the experimental classes will receive a gift worth 300 RMB.

The potential risks of the research

- The study will present no more than minimal risk to the participants.

- Please understand that your students'/teachers' participation are voluntary. They have every right to withdraw from the study at any time without negative consequences. All information related to your students'/teachers' will remain confidential and will be identifiable by codes known only to the researcher.

How results will be potentially disseminated

- This project will provide the participating kindergarten STEM curriculum-based teacher training and curriculum resources.

- Research results will be disseminated through thesis and journal article.

If you would like to obtain more information about this study, please contact ZENG Yue by email at [redacted]; Dr. Weipeng Yang by email at wyang@eduhk.hk.

If you have any concerns about the conduct of this research study, please do not hesitate to contact the Human Research Ethics Committee by email at hrec@eduhk.hk or by mail to Research and Development Office, The Education University of Hong Kong.

Thank you for your interest in participating in this study.

ZENG, Yue

November, 2022

香港教育大學

幼兒教育學系

參與研究同意書（學校）

插電式和非插電式編程課程對 5-6 歲兒童的計算思維、自我調節、心流體驗和自我效能感的影響

本幼稚園同意參與由楊偉鵬博士監督和 Alfredo Bautista 博士並由曾越執行的研究計劃。他/她們是香港教育大學幼稚教育系的教員/学生。

本人了解所收集的資料可能會用於未來的研究和學術發表，但本人有權保護本幼稚園學生和教師的隱私，個人資料不得外泄。

研究者已詳細解釋了相關程序和附帶的資料給本人。本人了解可能存在的風險。

本人自願讓本幼稚園的學生和教師參與這項研究。

本人理解本人和本幼稚園的學生和教師在研究過程中有權提出問題，並在任何時候決定退出研究，而不會對研究工作產生任何負面影響。

簽署:

園長姓名: _____

職位: _____

幼稚園名稱: _____

日期: _____

有關資料

插電式和非插電式編程課程對 5-6 歲兒童的計算思維、自我調節、心流體驗和自我效能感的影響

誠邀貴園參加楊偉鵬博士和 Alfredo Bautista 博士負責監督,曾越負責執行的研究計畫。他/她們是香港教育大學幼稚教育系的教員/学生。

研究計劃簡介

- 我們將為幼兒設計一系列插電和不插電編程活動，幫助幼兒學習順序、迴圈、條件、問題分解、調試等編程概念和技能。不插電的編程課程指的是沒有數字設備的編程教學，通常涉及紙和筆、卡片、貼紙書以及身體動作，而插電的編程課程指的是使用數字設備的編程教學。在插電的編程活動中，我們將使用 MBOLO 編程教具進行編程教學；在不插電的編程活動中，我們將使用無螢幕編程材料進行編程教學；
- 我們將為教師提供有關什麼是編程、如何開展編程活動的培訓；
- 我們將評估插電和不插電的編程課程對幼兒計算思維、自我調節能力、心流體驗、編程自我效能感的影響；
- 我們將對兩個實驗班進行的編程活動（大約兩個月）進行錄影；

- 在所有的編程活動結束後，我們將對實驗班的兩位老師和個別幼兒進行訪談，所有的訪談將被錄音。

研究方法

A)　工作及步驟

- 在編程課程開始前一周，實驗班的教師將在非工作的時間段接受2 小時的編程課程培訓。

- 編程活動（干預）將持續兩個月。每個星期兩次，每次大約 50 分鐘。實驗班的教師將承擔編程活動的教學工作（編程課程的具體內容詳見表 1）。

- 在編程課程結束之後，實驗班的教師將接受大約 60 分鐘的訪談。訪談將用於瞭解教師對幼兒編程教育的態度。訪談都將在貴校安靜的房間內進行。訪談將被錄音。

- 每個孩子的計算思維將在編程課程（干預）前後被評估（前測和後測）。計算思維評估通常需要 12 分鐘，由研究人員在幼稚園的一個安靜的房間裏進行。

- 每個孩子的自我調節能力將在編程課程（干預）前後被評估（前測和後測）。自我調節評估通常需要 15 分鐘，由研究人員在幼稚園的一個安靜的房間裏對幼兒進行。

- 每個孩子的流動體驗將在編程課程（干預）後被評估。 評估將需

要大約兩分鐘，由研究人員在幼稚園的一個安靜房間裏進行。

- 每個孩子的編程自我效能感將在編程專案（干預）後被評估。編程自我效能評估將由班級教師來完成。這將需要大約一個小時。

- 焦點小組訪談將分別與每個實驗組的 10 名兒童進行。訪談將被錄影，並將持續約一個小時。

表 1

編程課程

	插电式编程活动	活动目标	不插电编程活动	活动目标
Session 1	《認識 MOBLO 玩具》 《順序 1》	瞭解“順序”的概念；認識方向積木；能夠有順序地擺放方向積木，編寫科比打敗怪獸神的路線/程式。	《認識 unplugged programming 玩具》 《順序 1》	瞭解“順序”的概念；認識方向積木；能夠有順序地擺放方向積木，編寫奇奇旅遊的路線/程式。
Session 2	《順序 2》 《問題分解 1》	鞏固“順序”的概念；能夠較為熟練地運用方向積木編寫路線/程式。	《順序 2》 《問題分解 1》	鞏固“順序”的概念；能夠較為熟練地運用方向積木編寫路線/程式。

		在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。		在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。
Session 3	《順序 3》 《調試 1》	掌握“順序”的概念；能夠熟練運用方向積木編寫更長的路線/程式。 當編寫的路線/程式出現錯誤時，能夠檢查方向積木或記錄欄中的程式，找出錯誤的部分並糾正錯誤。	《順序 3》 《調試 1》	掌握“順序”的概念；能夠熟練運用方向積木編寫更長的路線/程式。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。

Session 4	《條件 1》 《表徵 1》	瞭解“條件”的概念；認識“動作積木”，瞭解在遇到特殊事件時需要使用動作積木；初步學習觀察路線，判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編寫科比打敗怪獸神的路線/程式。 觀察記錄欄中的符號，理解表徵的概念：使用抽象的符號來表示具體的指令。	《條件 1》 《表徵 1》	瞭解“條件”的概念；認識“條件指令卡”和工具積木，瞭解在遇到特殊事件時需要使用“條件指令卡”和工具積木；初步學習判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 觀察編程區的指令，理解表徵的概念：使用抽象的符號來表示具體的指令。

Session 5	《條件 2》 《問題分解 2》	鞏固“條件”的概念；能夠較為熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編寫科比打敗怪獸神的路線/程式。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。	《條件 2》 《問題分解 2》	鞏固“條件”的概念；能夠較為熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。
Session 6	《條件 3》 《調試 2》	進一步鞏固“條件”的概念；能夠熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木和動作積木，編	《條件 3》 《調試 2》	進一步鞏固“條件”的概念；能夠熟練地判斷在不同的情境下需要使用的不同工具，並正確使用方向積木、“條

		寫科比打敗怪獸神的路線/程式。 當編寫的路線/程式出現錯誤時，能夠檢查動作積木和方向積木或記錄欄中的程式，找出錯誤的部分並糾正錯誤。		件指令卡”和工具積木，編寫奇奇旅遊的路線/程式。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。
Session 7	《反復 1》 《表徵 2》	瞭解“反復”的概念；認識 NFC 積木和數字卡片；能夠找出簡單的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。	《反復 1》 《表徵 2》	瞭解“反復”的概念；認識反復積木；能夠找出簡單的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。

		觀察記錄欄中的符號，理解表徵的概念：使用抽象的符號來表示具體的指令。		觀察編程區的指令，理解表徵的概念：使用抽象的符號來表示具體的指令。
Session 8	《反復 2》 《問題分解 3》	鞏固“反復”的概念；能夠找出比較簡單的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。	《反復 2》 《問題分解 3》	鞏固“反復”的概念；能夠找出比較簡單的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。 在編寫程式的過程中，學會觀察起點和終點，並且能夠把多個步驟分解成一步一步。

Session 9	《反復 3》 《調試 3》	進一步鞏固“反復”的概念；能夠找出比較複雜的反復路線中的反復部分和反復次數，並利用 NFC 積木和數字卡片，輸入迴圈命令語。 當編寫的路線/程式出現錯誤時，能夠檢查方向積木和數字卡片或記錄欄中的程式，找出錯誤的部分並糾正錯誤。	《反復 3》 《調試 3》	進一步鞏固“反復”的概念；能夠找出比較複雜的反復路線中的反復部分和反復次數，並利用反復積木輸入迴圈命令語。 “機器人”幼兒在驗證路線的過程中發現錯誤時，“指令員”幼兒能夠檢查編程區的指令，找出錯誤的部分並糾正錯誤。
Session 10	《反復 4》	掌握“反復”的概念；能夠比較快速地找出複雜的反復路線中的反復部分和反復次數，並利	《反復 4》	掌握“反復”的概念；能夠比較快速地找出複雜的反復路線中的反復部分和

		用 NFC 積木和數字卡片，輸入迴圈命令語。		反復次數，並利用反復積木輸入迴圈命令語。
Session 11	《順序、條件和反復的綜合運用 1》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。	《順序、條件和反復的綜合運用 1》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。
	《演算法 1》	理解演算法的概念，並能運用順序、條件和反復等積木進行較為簡單的演算法設計。	《演算法 1》	理解演算法的概念，並能運用順序、條件和反復等積木進行較為簡單的演算法設計。

Session 12	《順序、條件和反復的綜合運用 2》 《演算法 2》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。 理解演算法的概念，並能運用順序、條件和反復等積木進行較為複雜的演算法設計。	《順序、條件和反復的綜合運用 2》 《演算法 2》	能夠綜合運用順序、條件和反復等積木編寫程式、解決較為簡單的問題。 理解演算法的概念，並能運用順序、條件和反復等積木進行較為複雜的演算法設計。

注：每個活動約需 50 分鐘。

B)　任何利益 (包括對參與者的補償)

- 教師將接受有關早期編程教育的免費培訓。

- 幼稚園將有機會與溫大研究團隊合作，將最先進的編程教育納入園本課程。

- 幼稚園和教師將學習如何尋找合適的課程材料和資源來設計編程活動。

- 實驗班的教師將收到價值 300 元的禮品一份。

參與期間可能面臨的風險與不適：

-這項研究對參與者的風險非常低。

-貴園學生/教師的參與完全是自願的。所有參與者在研究開始前或結束後都有權利選擇退出，並不會有任何不良後果。貴園學生/教師的相關資料將被保密，只有研究人員能夠讀取編碼後的資料。

研究結果的發佈方式：

-本項目將為參與的幼稚園提供有关编程教育的教師培訓和課程資源。

-研究結果將透過畢業論文和期刊論文來傳播。

如果您想獲得更多關於這項研究的資訊，請聯繫曾越（[redacted]）。

如果您對這項研究的研究倫理有任何意見，請隨時聯繫香港教育大學人類實驗

對象研究倫理委員會（電郵：hrec@eduhk.hk；地址：香港教育大學研究與發展事務處）。

謝謝您對參與這項研究的興趣。

曾越

2022 年 11 月 1 日