Computer Science Education

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ncse20

The efficacy of a computer science curriculum for early childhood: evidence from a randomized controlled trial in K-2 classrooms

Dandan Yang, Zhanxia Yang & Marina Umaschi Bers

To cite this article: Dandan Yang, Zhanxia Yang & Marina Umaschi Bers (10 Nov 2023): The efficacy of a computer science curriculum for early childhood: evidence from a randomized controlled trial in K-2 classrooms, Computer Science Education, DOI: 10.1080/08993408.2023.2279198

To link to this article: https://doi.org/10.1080/08993408.2023.2279198

COMPUTER SCIENCE EDUCATION
https://doi.org/10.1080/08993408.2023.2279198

The efficacy of a computer science curriculum for early childhood: evidence from a randomized controlled trial in K-2 classrooms

Dandan Yang, Zhanxia Yang and Marina Umaschi Bers

Boston College, Lynch School of Education and Human Development, Chestnut Hill, USA

ABSTRACT

Background and context: Despite the growing importance of computer science (CS) education, high-quality CS curricula for students in kindergarten to lower elementary grades are lacking. It is also unclear how students from underrepresented groups such as female students, students from low socioeconomic status, and students with disability respond to CS curriculum at this age range.

Objective: This study is aimed to examine the effectiveness of a novel CS curriculum (Coding as Another Language – ScratchJr) on the development of programming and computational thinking skills for students in kindergarten to second grade classrooms.

Method: We conducted a randomized controlled trial with 1057 students and used multilevel regression analysis to examine the impact of the CAL curriculum on students' coding and computational thinking skills, as well as moderation analysis to investigate how students' demographic characteristics including gender, socioeconomic status, English learners and disability interacted with the intervention effectiveness.

Findings: The CAL curriculum was effective in improving students' programming skills, but no significant differences were detected for students' computational thinking skills. Students with limited English proficiency and students from low socioeconomic backgrounds achieved similar gain in coding skills but students with disabilities and female students did not demonstrate the same improvements as their peers.

Implications: These findings shed light on the educational efficacy of CAL as a promising CS curriculum for young children and underscore the importance of understanding how underrepresented minority students respond to the curriculum in order to better guide the development and design of the CS programs.

ARTICLE HISTORY
Received 14 April 2023
Accepted 31 October 2023

KEYWORDS
computer science education; early childhood; elementary school; computational thinking; curriculum

Introduction

The field of computer and information technology (CIT) has seen significant growth in recent years, with increasing applications in various industries such as healthcare, finance, and transportation. According to data from the federal Bureau of Labor Statistics (BLS), in

CONTACT Dandan Yang dandan.yang.2@bc.edu Boston College, Lynch School of Education and Human Development, 140 Commonwealth Ave, Chestnut Hill 02467, USA

Supplemental data for this article can be accessed online at https://doi.org/10.1080/08993408.2023.2279198

2018, the median annual salary for CIT-related occupations was $86,320, significantly higher than the median salary of $38,640 for all occupations (Wurman & Donovan, 2020). The BLS projects a 12% growth in the CIT industry from 2018 to 2028, resulting in the creation of half a million new jobs, far surpassing the average for all occupations (Wurman & Donovan, 2020).

As the demand for computer and information technology professionals continues to rise in the digital economy, educators, parents, researchers, and policymakers have placed a greater emphasis on computer science (CS) education in the K-12 education system. In 2016, former President Obama introduced the Computer Science for All initiative, with the goal of providing all students with hands-on computer science and math classes that would prepare them for job opportunities upon graduation (Computer Science for All, 2016). This initiative, along with similar efforts at the state and local level (Code.org Advocacy Coalition, 2017; National Science Board, 2018), aims to ensure that students have access to the computational skills and knowledge necessary to succeed in an increasingly digital world. Furthermore, CS education can provide students with critical thinking, problem-solving, and creativity skills which are not only valuable in the tech industry but also in various other fields (Papert, 1980; Wing, 2006). The increasing availability of CS education in K-12 schools provides students with the opportunity to explore potential career paths, prepare them for higher education, and equip them with the skills required to succeed in the 21st century.

One concern of current CS programs is that most are designed for older children, such as middle school students and above, disregarding the fact that early childhood is a critical period for children's development. Early childhood learning provides the opportunity for children to acquire foundational learning skills, cognitive abilities, and social-emotional skills (Friedman-Krauss et al., 2021; Ruhm & Waldfogel, 2012). Research shows that well-designed early childhood educational programs and interventions can have both short-term and long-term positive impacts on children's social competence, intellectual, and academic performance. For instance, a study by Friedman-Krauss et al. (2021) found that high-quality preschool programs can have significant positive effects on children's academic and social-emotional development, including improved school readiness, higher test scores, and reduced special education placements. Most early childhood education programs, however, focus on language arts and mathematics, with few including CS, a subject also suitable for early childhood education. Well-designed CS programs can provide a learning environment where young children can "play to learn while learning to play" (Bers et al., 2018, p. 146). This approach aligns with the way young children learn and allows them to develop computational thinking (CT) skills, problem-solving skills, creativity, and socio-emotional skills in a fun and engaging way (Sarama & Clements, 2009; Papert, 1980).

The K-12 Computer Science Framework (undefined) recommends that by the end of second grade, students should have a basic understanding of key concepts such as software, algorithms, variables, control structures, and modularity, as well as skills such as debugging. However, the lack of high-quality and developmentally appropriate CS programs for early childhood education makes it challenging for teachers to effectively integrate CS into their existing curriculum and for students to acquire the necessary CS and CT skills by the end of the second grade. This, in turn, hinders their ability to be better prepared for more advanced CS curricula in later stages of their education.

This is problematic as learners from historically disadvantaged groups, such as students with limited English proficiency (referred to as LEP in this paper, Jacob et al., 2022), students with disabilities (Bouck & Yadav, 2022; Israel et al., 2015), students from low socioeconomic status (SES, Chachashvili-Bolotin et al., 2016), and female students (Cheryan et al., 2015), are underrepresented in the STEM fields. This lack of diversity not only limits the pool of potential talent but also perpetuates stereotypes and biases that can further discourage underrepresented groups from pursuing STEM careers. Thus, it is important to understand how students from diverse backgrounds respond to CS programs in order to guide future instruction, research, and policy.

CAL is a CS program for K-2 students developed by the CAL Research Group at Boston College (Bers et al., 2023). This curriculum introduces the powerful ideas from CS integrated with literacy in a developmentally appropriate, structured, creative, and playful way. The CAL curriculum includes 24 lessons with unplugged and plugged activities using the most popular programming language ScratchJr among young children across the world (Bers, 2018; Bers & Sullivan, 2019; Flannery et al., 2013; Portelance et al., 2015). Although early evidence indicates the promise of this curriculum (Bers et al., 2023), there are no experimental research studies that systematically examine the educational efficacy of this program on the coding and CT skills for children from kindergarten to second grade.

To this end, the current study fills the gaps in the literature by examining the effectiveness of the CAL curriculum in children's programming and CT skills through a randomized controlled trial. We also explore how economically, linguistically, and physically underrepresented students interact with the curriculum. As the first study to systematically examine the efficacy of the CAL curriculum for children in K-2 grades through experimental design, our results will yield critical evidence about the educational effectiveness of the CAL program. The potential findings on how underrepresented students (e.g. those with low SES, English learners, and those with disabilities) performed during the intervention highlight implications for educators and researchers in developing and improving early childhood CS programs to better meet the needs of the underrepresented student body and promote inclusive excellence in STEM education. The following research questions guided our study design and data analysis:

(1) What is the impact of the CAL curriculum on children's coding skills?
(2) What is the impact of the CAL curriculum on children's CT skills?
(3) How are child characteristics (grade level, socioeconomic status, gender, disability status, language status) and the number of lessons completed associated with the effectiveness of the CAL intervention on both children's coding skills and CT skills?
(4) How do child characteristics moderate the effectiveness of the CAL intervention on children's coding skills and CT skills?

Computer programming and computational thinking

In recent years, the focus on CS education has expanded from teaching computer programming to fostering higher-level cognitive and analytical skills, known as computational thinking (Fayer et al., 2017, US Department of Education, Office of Educational

Technology, 2017; Wing, 2006, 2011). Jeannette Wing first defined "computational thinking" as a "fundamental skill for everyone" that involves "solving problems, designing systems, and understanding human behaviors, by drawing on the concepts fundamental to computer science" (Wing, 2006).

Drawing from the constructionist learning theory (Papert, 1980), some leading researchers in CS education view CT as a vehicle for creation and expression through applications and artifacts that are relevant and meaningful to individuals and are shareable with social networks (Brennan & Resnick, 2012; Bers, 2022; Kafai & Burke, 2014). In this framework, learning to program using a particular language allows children to develop their abstract thinking, logical reasoning, and problem-solving skills, as well as engage in creative design processes, bringing programs and applications of personal relevance and self-expression.

One criticism of current CS programs is that most focus more on teaching children how to code using a programming language, while failing to incorporate CT training into their curriculum (Fayer et al., 2017). Some researchers argue that this narrow focus on coding can lead to a lack of understanding of the broader concepts of CS and the ability to apply these concepts to problem-solving in other fields. Recognizing this limitation, some researchers in CS education started to develop age-appropriate curriculums that go beyond coding and incorporate activities targeting higher-level cognitive abilities, such as CT and problem-solving (Bell & Lodi, 2019). For example, a CS curriculum for PreK to second-grade students (CAL-KIBO, Bers, 2022) was developed using the KIBO robot, which introduces powerful ideas from CS, in combination with literacy in a playful, structured, and developmentally appropriate way. Evidence from an experimental study has shown that the CAL-KIBO curriculum was effective in improving both coding skills measured by Coding Stages Assessment-KIBO and unplugged CT skills measured by TechCheck for first and second graders (Bers, 2022).

Underrepresented groups in STEM education

Significant evidence indicates that inequality exists in the current K-16 STEM education (e.g. Baber, 2015; McGee, 2021; Xie et al., 2015). Researchers and educators have pointed out that most of the existing STEM curricula are not inclusive or, in other words, do not adequately represent the diversity of the student population (National Science Board, 2018). Students from historically underrepresented groups, such as female students (Cheryan et al., 2015), students from low SES backgrounds (Chachashvili-Bolotin et al., 2016; Su et al., 2023), students with limited English proficiency (Jacob et al., 2022), and students with disabilities (Bouck & Yadav, 2022; Israel et al., 2015), tend to have more limited access to resources, face extra challenges, and are less likely to succeed in STEM education starting from early childhood.

Specifically, research has pointed out that socioeconomic disparities universally exist in STEM education, starting as early as preschool and early elementary years (Betancur et al., 2018). Both household income and parental education contribute to the gaps in science achievement through limited access to informal science learning opportunities both inside and outside of the home environment (Betancur et al., 2018). These science learning opportunities include but are not limited to advanced STEM courses and programs, as well as opportunities for extracurricular activities and competitions related to

STEM (Betancur et al., 2018; Blums et al., 2017). Moreover, their parents are also less likely to get involved in and provide support for their children's education (Milner-Bolotin & Marotto, 2018; Thomas et al., 2020), which includes not only STEM education but also other subjects such as math and literacy. Betancur and colleagues found that lower achievement in math and literacy mediates the relationship between low SES and students' disadvantaged STEM education outcomes (Betancur et al., 2018).

Female students in STEM education tend to face societal stereotypes and gender bias. Studies have shown that girls are often exposed to negative stereotypes about their abilities in STEM subjects and are less likely to be encouraged to pursue careers in STEM fields (Cheryan et al., 2015; Jiang et al., 2020). These stereotypes can lead to girls undervaluing their abilities and having lower expectations for themselves in STEM subjects.

Similar patterns have been observed for students with limited English proficiency and those with disabilities, including but not limited to any mental and physical impairment or developmental delays. One of the main barriers that children with disabilities face in STEM education is a lack of access to accommodations, such as assistive technology (Hwang & Taylor, 2016; Klimaitis & Mullen, 2021). Due to their disability, these students are sometimes not included in general education STEM classes, or even if they receive instruction in general education classrooms, the teaching methods and pedagogy may not necessarily be appropriate for their needs (Hwang & Taylor, 2016; Klimaitis & Mullen, 2021).

Additionally, students with limited English proficiency and English language learners may struggle with understanding STEM content presented in English, which can impede their ability to fully engage in STEM learning and activities (Buxton & Lee, 2014; Jacob et al., 2022). They may have difficulty understanding key concepts and vocabulary, making it challenging for them to participate in class discussions and complete assignments (Buxton & Lee, 2014). This can result in a lack of engagement and motivation in STEM subjects, negatively impacting their performance and achievement.

In conclusion, given the aforementioned equity issues, it is important for researchers and educators to understand how students from underrepresented groups perform and respond to STEM programs compared to their peers. This information is crucial for adjusting and improving the current curriculum to better suit their needs and for developing programs that support inclusive excellence for all students.

The CAL curriculum

Guided by the pedagogical premise that coding is a new literacy of the 21st century, the CAL Curriculum was developed by the CAL Research Group over a three-year period using a design-based research approach (Cobb et al., 2003). The CAL curriculum incorporates literacy instruction in teaching coding, with the ultimate goal of nurturing global citizens with powerful intellectual tools, including literacy, math, and programming, to verbalize, express, innovate, and thrive in the 21st century.

The CAL curriculum was developed guided by the Positive Technological Development framework (Bers, 2006, 2012). It includes classroom activities that engage children in a coding playground by providing opportunities to design and code their own personally meaningful projects, interact with others, and grow socially and emotionally by learning how to use a new language (a.k.a ScratchJr) for expressive and communicative purposes (Bers et al., 2023). The 24 lessons in the curriculum are structured around six powerful ideas from both CS (*algorithms, design process, representation, debugging, control structures, modularity, and hardware/software*) and literacy (*sequencing, the writing process, alphabet and letter-sound correspondence, editing and audience awareness, literary devices, phonological awareness, and tools of communication and language*) respectively. More details about the six powerful ideas, as well as the lessons associated with them in the first-grade curriculum, can be found in Table 1.

The ScratchJr app is utilized in the CAL Curriculum as the coding language and platform. ScratchJr, co-developed by the Devtech Research Group led by Marina Umaschi Bers, the LifeLong Kindergarten group led by Mitchel Resnick, and the PICO company, is an introductory programming language designed for children aged 5–7 to create their own stories and games (Portelance et al., 2015). With a graphical programming environment, children can create animated and interactive scenes and stories while engaging in age-appropriate problem-solving and computer programming. As a free downloadable app used by over 38 million children worldwide, ScratchJr has been shown to be developmentally appropriate for young children with low floor and high ceiling features (Flannery et al., 2013; Stamatios, 2022; Unahalekhaka & Bers, 2021).

Building on ScratchJr, the CAL Curriculum consists of 24 lessons (18 hours of instruction) and has three adaptive versions for kindergarten, first grade, and second grade, respectively. As the grade level increases, although the number of lessons and the total instruction time remain the same, the levels of difficulty and complexity of the class activities and teaching materials increase accordingly (see Supplementary). For each lesson, multiple classroom activities guide and allow children to explore the powerful ideas of coding and literacy in a collaborative, creative, and playful manner. These classroom activities include warm-up activities, opening & closing circles, structured challenges, expressive collaborations, unplugged time, word time, and more. For example, in lesson 6 of the first-grade curriculum, students will learn the *Repeat Forever block* and

Table 1. Powerful ideas from computer science and literacy in CAL.

Powerful Ideas from Computer Science	Powerful Ideas from Literacy	Connecting the Powerful Ideas
Algorithms (Lessons 2, 5, 6, 7, 9, 17, 19, 20, 22, 23, 24)	Sequencing (Lessons 2, 5, 6, 7, 9, 19, 20, 24)	A set of instructions or steps for solving a specific problem or achieving a specific task
Design Process (Lessons 5, 6, 7, 8, 10, 11, 19, 20, 21, 22, 23, 24)	Writing Process (Lessons 7, 8, 10, 11, 19, 21, 22, 23, 24)	Creative, iterative, cyclic processes that involve imagining, planning, making, revising, and sharing
Representation (Lessons 1, 3, 4, 10)	Alphabet and Letter-Sound Correspondence (Lesson 4)	Symbols with various attributes as methods of representation
Debugging (Lessons 10, 11, 12, 22)	Editing and Audience Awareness (Lessons 6, 10, 11, 12)	Systematic analysis, testing, and evaluation to improve communication to the intended audience
Control Structures (Lessons 14, 15, 16, 17, 18)	Literary Devices (Lessons 14, 15, 16, 17, 18)	fundamental building blocks for expressing a group of thoughts
Modularity (Lessons 20, 21)	Phonological Awareness (Lessons 5, 6, 12, 20, 21)	Decomposition, or breaking down a complex task into smaller tasks.
Hardware and Software (Lessons 3, 13)	Tools of Communication and Language (Lessons 1, 3, 13, 17)	Communicating abstract ideas through tangible means.

Note: *Literary devices are referred to the specific techniques and tools used by writers to create a more engaging and impactful piece of literature.* .

the *Pop block* in ScratchJr, while exploring the powerful ideas of algorithms and design processes in CS, as well as the powerful ideas of *Sequencing, Phonological Awareness, Editing, and Audience Awareness* in literacy. The following classroom activities are integrated into this lesson:

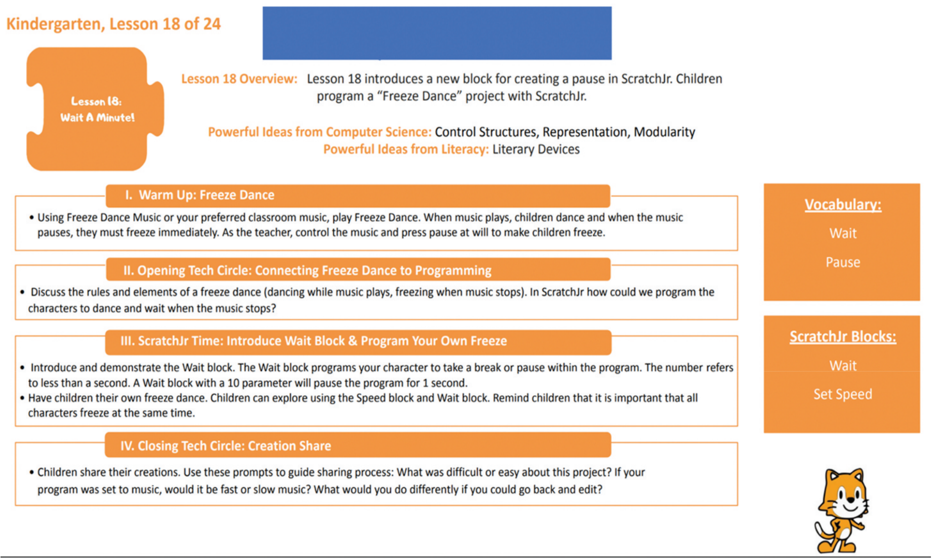
- Warm-up activity where the students dance the Hokey-Pokey (5 min);
- Opening tech circle where the teacher will explain to the children that they will be revising their Hokey-Pokey programs by adding more blocks and finishing their programs. (5 min)
- Structure challenge where the teacher will introduce the Repeat Forever and Pop blocks to the students (10 min)
- Expressive explorations where students are encouraged to continue to code their Hokey-Pokey using the two new blocks (15 min)
- Closing tech circle where the students will share their projects in small groups, explain how they improved their programs (10 min)

A visual presentation of a CAL lesson in the kindergarten curriculum can be found in Figure 1, and a more detailed lesson plan can be found in Supplementary. Below (see Figure 2) is a picture of the children working on a programming project using ScratchJr during a CAL lesson.

Methods

Study procedures

To evaluate the impact of the CAL curriculum on students' coding skills and CT skills, a randomized control trial with a delayed treatment was conducted among 26 public



Kindergarten, Lesson 18 of 24

[redacted]

Lesson 18:
Wait A Minute!

Lesson 18 Overview: Lesson 18 introduces a new block for creating a pause in ScratchJr. Children program a "Freeze Dance" project with ScratchJr.

Powerful Ideas from Computer Science: Control Structures, Representation, Modularity
Powerful Ideas from Literacy: Literary Devices

I. Warm Up: Freeze Dance

- Using Freeze Dance Music or your preferred classroom music, play Freeze Dance. When music plays, children dance and when the music pauses, they must freeze immediately. As the teacher, control the music and press pause at will to make children freeze.

II. Opening Tech Circle: Connecting Freeze Dance to Programming

- Discuss the rules and elements of a freeze dance (dancing while music plays, freezing when music stops). In ScratchJr how could we program the characters to dance and wait when the music stops?

III. ScratchJr Time: Introduce Wait Block & Program Your Own Freeze

- Introduce and demonstrate the Wait block. The Wait block programs your character to take a break or pause within the program. The number refers to less than a second. A Wait block with a 10 parameter will pause the program for 1 second.
- Have children their own freeze dance. Children can explore using the Speed block and Wait block. Remind children that it is important that all characters freeze at the same time.

IV. Closing Tech Circle: Creation Share

- Children share their creations. Use these prompts to guide sharing process: What was difficult or easy about this project? If your program was set to music, would it be fast or slow music? What would you do differently if you could go back and edit?

Vocabulary:
Wait
Pause

ScratchJr Blocks:
Wait
Set Speed

Figure 1. A visual presentation of a CAL lesson in kindergarten curriculum.

Figure 2. A picture of a teacher working with the students in a CAL classroom.

elementary schools in two sites on the East Coast of the United States. The CAL curriculum was developed for kindergarten, first, and second-grade students, so only these three grade levels were included in the study. At the beginning of the study, the participating schools underwent a randomization process and were assigned to either the treatment group, where the curriculum was implemented immediately, or the control group, where the curriculum was implemented in the following school year. The randomization occurred at the school level, meaning that all the participating classrooms in one school were assigned to the same treatment condition.

During the randomization, a stratified random sampling technique was employed to obtain representative and comparable samples in demographics and grade levels for the treatment and control groups. Specifically, in site A, schools were randomly assigned within three strata identified based on the poverty quartile, while in site B, schools were randomly assigned within three strata formed based on the number of participating classrooms within the three grade levels. This resulted in 13 schools (7 from site A and 6 from site B) in the treatment condition and 13 schools in the control condition (6 from site A and 7 from site B).

Before the intervention began, all students underwent the Coding Stages Assessment (CSA) and TechCheck from the late Fall semester of 2021 to the beginning of the Spring semester of 2022. During the study, students in the treatment group received the CAL curriculum intervention during the Spring semester of 2022, while the students in the business-as-usual control group did not receive any intervention beyond their regular school curriculum. Post-tests of CSA and TechCheck were administered to all the students before the end of the Spring semester of 2022. The pre-test was administered 0–6 weeks before the start of the coding intervention, and the posttest 0–3 weeks after the last lesson of the coding intervention. The fidelity of the CAL curriculum implementation for the treatment group was tracked by collecting teachers' lesson logs, post-implementation surveys, and teacher interviews. Due to the pandemic, researchers were not allowed to conduct classroom visits, so information about treatment fidelity was self-reported by the teachers. Similarly, the research team did not have access to information regarding the regular school curriculum for both groups and how the CAL curriculum was incorporated into the existing curriculum in the treatment schools. It is worth noting that site B (accounting for approximately 20% of the whole sample) started the intervention two months later than site A due to administrative reasons. Therefore, while most of the classrooms completed the entire curriculum in site A, the average number of lessons completed in site B was only approximately 9, ranging from 4 to 20 total completed lessons.

Before the experiment began, consent forms were collected from all participating teachers and the students' parents by our site coordinators. The Institutional Review Board (IRB) approval was obtained at the authors' university. Consent forms were gathered from a total of 1507 students, and the analytical sample of the current study included 1057 students who completed the intervention with pre and post assessments. Attrition was due to various reasons: (1) Implementation constraints during the COVID-19 pandemic, including safety concerns, sick leave, and remote assessment procedures. (2) Teachers not completing the intervention due to job changes, illness, or maternity leave. (3) Missing pre-test or post-test scores due to technical issues or administrative reasons.

Teacher professional development training

All the teachers in the treatment group participated in professional development (PD) training hosted by an experienced trainer (Kapoor et al., 2023). The PD training was divided into two parts, each lasting a total of four hours: the introduction of programming with ScratchJr and the CAL curriculum. Teachers engaged in these two 2-hour synchronous PD sessions with an experienced trainer via Zoom. The first part began with introductions, followed by an introduction to ScratchJr. This was succeeded by guided explorations, a brief break, an overview of advanced ScratchJr features, a session on recreating a story using ScratchJr, sharing their created projects, and closing with a Q&A session. The second part included an exploration of four powerful metaphors (coding as a playground, coding as another language, coding as a bridge, and coding as a palette of virtues), followed by an introduction to the CAL curriculum. This was followed by a brief break, an in-depth exploration of one lesson, reflection time, a discussion on the logistics of the research study, and closing with another Q&A session.

Participants

In total, the analysis sample included 237 students in kindergarten, 348 students in the first grade, and 472 students in the second grade. Female students accounted for 51.37% of the entire sample. There were 124 students who received an Individualized Education Plan, and 127 students were identified as having Limited English Proficiency. A total of 248 students qualified for free or reduced lunch. Regarding racial ethnicity, the majority of

Table 2. Demographic information for the students in the treatment and control group.

	Treatment		Control	
	N	Percentage	N	Percentage
Gender (Female)	279	50.82%	264	51.97%
Ethnicity				
White	341	62.11%	334	65.75%
Hispanic	102	18.58%	77	15.16%
African American	37	6.74%	57	11.22%
Asian	29	5.28%	16	3.15%
Other	34	6.19%	24	4.72%
Grade Level				
Kindergarten	141	25.68%	96	18.90%
Grade 1	153	27.87%	195	38.39%
Grade 2	255	46.45%	217	42.72%
Disability	61	11.11%	63	12.40%
Limited English Proficiency	77	14.03%	50	9.84%
Low SES	130	23.68	118	23.23

students were Caucasian (63.86%), followed by Hispanic (16.93%) and African American (8.89%). Only 4.26% of the students were Asian, and 5.49% were of mixed race or from other ethnicities. Detailed student demographic information for the treatment and control groups can be found in Table 2.

Measures

Coding stages assessment

The Coding Stages Assessment (CSA) is a validated assessment (de Ruiter & Bers, 2021) used to evaluate coding skills using the ScratchJr programming language for children in kindergarten through lower elementary grades. During the assessment, the child is prompted with a series of questions and either responds verbally or solves a coding task using ScratchJr. The CSA consists of 25 question items distributed across five stages: *Emergent, Coding and decoding, Fluency, New knowledge, and Purposefulness*. In each stage, if the child fails to answer more than two questions correctly, the test ends after the last question of that stage; otherwise, the test continues until completion.

Due to the pandemic, the CSA was administered remotely via Zoom. Trained research assistants scheduled Zoom meetings with the participants and administered the CSA on a one-on-one basis while the students were at school. Students did not receive any assistance from either their teachers or parents during the assessment administration. Each CSA question was originally scored as 1 for a correct answer and 0 for an incorrect answer, with the scores then weighted based on the difficulty level of the questions. The split-half reliability for the CSA is Guttman’s Lambda 6 = .94(de Ruiter & Bers, 2021).

TechCheck

As the measure of computational thinking in this study, TechCheck has been validated and used among kindergarten, first, and second graders (Relkin et al., 2020). The TechCheck is an unplugged assessment that includes 15 multiple-choice questions addressing six domains of computational thinking (algorithms, control structure, debugging, hardware/software, modularity, representation). Example questions are provided in Supplementary. In the present study, the assessment was administered using computers and tablets, but the format of the assessment, not the mode of administration, is what makes it unplugged. The child answers the prompts by selecting one of four options. Each correct answer earns one point, with a maximum score of 15 points. The assessment begins with two practice questions to familiarize students with the format, but these are not included in the scoring. All questions must be answered to complete the assessment. The TechCheck assessment typically takes about 13 minutes for children to complete.

Analysis

To answer our research questions, we conducted descriptive statistical analysis and correlational analysis to gain a comprehensive understanding of the variables, as well as the relationship between them. Considering the nested nature of the data, a series of multilevel linear regression analyses was conducted to establish baseline equivalence and examine the impact of the CAL curriculum on the CSA (models 1, 2, and 3) and TechCheck post-tests (models 5, 6, and 7). We used a two-level mixed effects linear regression with schools at level 2 and students at level 1, recognizing that students were nested within schools and that teaching assignments and administrations within schools varied significantly. We created four moderation terms with the demographic measures and the intervention condition (e.g. gender*condition) to investigate how students' gender, SES, LEP, and disability status (those who qualified for the Individualized Education Plan, IEP) moderated the learning outcomes measured by the CSA post-test (model 4) and TechCheck post-test (model 8). Additionally, we controlled for the number of lessons completed in models 3, 4, 7, and 8 to account for the potential influence of treatment validity on the results. All data analysis was conducted using R Studio version 2022.07.2.

Results

Table 3 summarizes the descriptive statistics of the CSA and TechCheck pre and post scores for the students in the treatment and control groups. Specifically, the CSA scores increased from 3.78 (SD = 2.89, Min = 0, Max = 26) to 11.43 (SD = 6.56, Min = 1.1, Max = 39) for the students in the treatment group, and from 3.51 (SD = 2.19, Min = 0, Max = 13.9) to 5.58 (SD = 3.28, Min = 0, Max = 26) for the students in the control group. For TechCheck, students in the treatment group scored an average of 7.31 (SD = 2.52, Min = 1, Max = 15) at pre-test and 8.69 (SD = 2.67, Min = 1, Max = 15) at post-test, while students in the control group scored an average of 7.32 (SD = 2.41, Min = 0, Max = 14) at pre-test and 8.71 (SD = 2.51, Min = 2, Max = 15) at post-test.

Table 3. Descriptive statistics of the assessment scores.

	Treatment (549)				Control (508)			
	Mean	SD	Min	Max	Mean	SD	Min	Max
CSA Pre-test	3.78	2.89	0.00	26.00	3.51	2.19	0.00	13.90
CSA Post-test	11.43	6.56	1.10	39.00	5.58	3.28	0.00	26.00
CSA Gain	7.65	5.89	−5.80	32.10	2.07	2.66	−3.40	15.20
TechCheck Pre-test	7.31	2.52	1.00	15.00	7.32	2.41	0.00	14.00
TechCheck Post-test	8.69	2.67	1.00	15.00	8.71	2.51	2.00	15.00
TechCheck Gain	1.38	2.49	−5.00	7.00	1.38	2.40	−5.00	7.00

Table 4. Fixed effects of the multilevel regression models predicting the CSA post-test.

	Model 1	Model 2	Model 3	Model 4
Intercept	5.83***	2.01***	3.57***	2.96***
Treatment Condition	5.52***	4.81***	1.33	2.51**
CSA Pre-test		1.03***	0.90***	0.90***
Grade – 2			0.37	0.40
Grade - K			−1.97***	−1.97***
Limited English Proficiency (LEP)			−0.87	−0.30
Gender			−0.72**	−0.13
Individualized Education Program (IEP)			−1.40**	−0.43
Free or Reduced Lunch (SES)			−0.63	−0.37
# Lessons Completed			0.23***	.23***
IEP*Condition				−1.93*
LEP*Condition				−.97
Gender*Condition				−1.16*
SES*Condition				−0.45
N	1057	1057	1051	1051

Note: * *p*<.05, ** *p*<.01, *** *p*<.001. .

We first conducted unconditional multilevel regression models on students' CSA and TechCheck pre-test scores with school ID as the nested level. The Intraclass Correlation Coefficient (ICC) was .27 for the CSA pre-test model and .05 for the TechCheck pre-test model. This suggests that 27% of the variance in CSA pre-test scores and 5% of the variance in TechCheck pre-test scores are related to the schools that the students were nested in, indicating the preference for multilevel modeling.

To establish baseline equivalence, conditional multilevel models were conducted with individual students being level 1 and schools being level 2. Our results showed that at pre-test, the CSA scores did not differ significantly between the treatment group and control group ($\beta = .81$, $SE = .59$, $p = .18$), although descriptive statistics indicate that the treatment group, in general, scored 0.3 higher than the control group. Similarly, we did not detect significant between-group differences for the TechCheck pre-test scores ($\beta = .01$, $SE = .30$, $p = .98$).

RQ1: what is the impact of the CAL curriculum on children's coding skills?

Following similar procedures, we ran a multilevel regression analysis (Model 1) with the individual students being level 1 and schools being level 2. We included the treatment condition as a level 1 variable to investigate whether there were any between-group differences for CSA post-test scores. Model 1 suggests that students in the treatment group, in general, scored 5.52 points higher than the students in the control group ($p < .001$). The effect size for this impact is large with a Cohen's $d = 1.22$. Considering the variance in the pre-test CSA scores, we ran a second model controlling for students' pre-test scores. Model 2 shows that, after controlling for students' pre-test scores, the treatment group scored 4.81 points higher than the children in the control group ($p < .001$). Details of the fixed results can be found in Table 4.

RQ2: what is the impact of the CAL curriculum on children's CT skills?

Similar analysis procedures were taken to answer the second research question. The results from model 5 indicated that the CAL curriculum did not have a significant impact on children's CT skills, measured by TechCheck ($\beta = -.11$, p

Table 5. Fixed effects of the multilevel regression models predicting the TechCheck post-test.

	Model 5	Model 6	Model 7	Model 8
Intercept	8.51***	4.64***	5.60***	5.56***
Treatment Condition	−0.11	−0.08	−0.46	−.41
TechCheck Pre-test		0.54***	0.51***	0.51***
Grade – 2			−0.24	−0.23
Grade - K			−0.66**	−0.64**
Limited English Proficiency (LEP)			−0.11	0.01
Gender			−0.48***	−0.28
Individualized Education Program (IEP)			−0.84***	−0.72*
Free or Reduced Lunch (SES)			−0.44*	−0.77**
# Lessons Completed			.03	.03
IEP*Condition				−0.27
LEP*Condition				−0.20
Gender*Condition				−0.43
SES*Condition				0.64
N	1057	1057	1051	1051

Note: ** p<.05, ** p<.01, *** p<.001..*

= .77). And the null effect of the intervention remained consistent when controlling for students' TechCheck pre-test ($\beta = -.08$, $p = .75$). Model 6 also suggested that students who scored higher on the TechCheck pre-test tended to perform better on the TechCheck post-test ($\beta = .54$, $p < .001$). Details of the fixed effect results can be found in Table 5.

RQ3: how are child characteristics and the number of lessons completed associated with the effectiveness of the CAL intervention on both children's coding skills and CT skills?

Analysis results of the multilevel regression model on CSA post-test (Model 3) shows that controlling for the intervention condition and students' pre-test scores, second graders performed similarly to children in the first grade ($\beta = .37$, $p = .27$), while students in kindergarten in general scored 1.97 points lower than their first-grade peers ($p < .001$). We also found that regardless of the treatment condition, girls performed worse than boys ($\beta = -.72$, $p = .01$), and students with disabilities performed worse than their typical developing peers ($\beta = -1.41$, $p < .001$).

For TechCheck, Model 7 shows that regardless of the intervention condition and their pre-test scores, girls ($\beta = -.48$, $p < .001$), students with disabilities ($\beta = -.84$, $p < .001$) and students from low SES backgrounds ($\beta = -.44$, $p = .01$) scored significantly lower than their counterparts on the TechCheck post-test. Similar to the CSA, kindergarten students scored significantly lower than the first graders' TechCheck scores ($\beta = -.66$, $p = .001$).

In addition, we found that the treatment fidelity, measured by the number of lessons completed by the teachers, significant predicted students' CS post-test scores ($\beta = .23$, $p < .001$) but did not yield a significant impact on students' TechCheck post-test scores ($\beta = .03$, $p = .11$).

RQ4: to what extent do child characteristics moderate the effectiveness of the CAL intervention on children's coding skills and CT skills?

We also conducted a moderation analysis to investigate how students' demographic characteristics interact with the effectiveness of the CAL curriculum. Based on the results of Model 4, students with disabilities in general scored 1.93 points lower than their non-disabled peers ($p = .02$) and girls in general scored 1.16 points lower than boys ($p = .03$) in response to the CAL curriculum. No statistically significant differences were found for children with limited English proficiency and those from low SES backgrounds.

For TechCheck scores, based on the results of Model 8, we did not find any statistically significant differences in the post-test scores for students with different characteristics. However, it is worth noting that there is a borderline effect for students from low SES backgrounds. Students from low SES families scored .64 points higher than their wealthier peers ($p = .07$).

Discussion

The effectiveness of the CAL curriculum on students' coding and computational thinking

While educators and researchers are increasingly paying more attention to CS education, most of the existing CS programs are designed for older children and teenagers, and high-quality, evidence-based CS curricula for young children are lacking (Bers & Sullivan, 2019). To the best of our knowledge, our study is the first experimental study that provides evidence that the CAL curriculum was effective in improving the programming skills for students from kindergarten to second grade. These findings are consistent with the positive findings from the previous pilot studies on the CAL-related curriculum (Bers et al., 2023). It is worth noting that not all classrooms participating in the study completed all the 24 lessons in the curriculum due to the constraints created by COVID-19. This implies that the effect sizes of the CAL curriculum generated in the current study might be smaller than the real effect size if all 24 lessons in the curriculum are implemented. Analysis results from Model 3 also indicated that the number of lessons completed by the teacher significantly predicted students' coding skills at the post-test. Future research is warranted to replicate the results and further investigate the effect sizes of the CAL when all lessons are implemented.

In addition, a larger effect size on the CSA was found for students in the first and second grades compared to kindergarten. This is probably due to the fact that the CAL curriculum is adaptive to grade levels, meaning that the curriculum for first and second grades is designed to be more challenging than that for the kindergarteners. For example, the most advanced ScratchJr skill introduced in the kindergarten curriculum is adjusting parameters, while in the second grade, students also learned more advanced skills such as parallel programming, next repeat loops, and sending messages using multiple colors. It is entirely possible that the students in kindergarten did not learn the advanced coding blocks during the intervention that were tested in the CSA. Future research should examine whether students in kindergarten learn as much as the students in the first and second grade if they were given the same curriculum. Another possible but less likely reason for this large effect size is that students in elementary school are more cognitively and mathematically ready to learn coding than children in kindergarten. Evidence suggests that early childhood is a key developmental period characterized by rapid cognitive development (Campbell et al., 2001), and educational investment in early childhood education, particularly pre-K and kindergarten programs, such as the Head Start, has the highest economic value compared with programs at higher grade levels (Garces et al., 2002). However, most of these studies only focused on language arts and cognitive capability benchmarks. Coding is more cognitively demanding compared to most of the early language arts instructions and it also relies on math skills such as counting, addition, subtraction, and multiplication, which are often not introduced until elementary grades. Future studies need to be conducted to examine in further detail how children in different grade levels respond to the curriculum in the classroom to better guide future curriculum design and policy.

Conversely, we did not find any significant differences in students' CT scores. As previously stated in this paper, compared with coding and programming, CT involves higher-level cognitive and metacognitive skills, which may require longer exposure to elicit growth and development (García Peñalvo et al., 2016). Although various aspects of CT, such as abstract thinking, logical reasoning, and problem-solving, could be developed while learning to code, it does not necessarily mean that students who are proficient in coding using a specific programming language can always be successfully transferred to significant improvements in their CT skills.

The CAL curriculum was designed to have 24 lessons, each lasting 45 minutes. However, this study was carried out during the COVID-19 pandemic under many constraints, and due to the tight schedule of some schools, not all participating teachers completed the 24 lessons using 45-minute blocks. According to the lesson logs and interviews from the teachers, some of the classrooms, especially those from site B, only implemented less than half of the lessons before the post-tests. This level of intervention intensity may be enough to improve children's coding skills using the ScratchJr app but may not be sufficient for developing higher-level cognitive skills related to CT.

The second explanation for this null finding is that although the students in the control group did not receive the CAL curriculum, they may have been exposed to other unplugged CS curricula that were regularly implemented in their school. This may have led to a natural growth in computational thinking skills for the control group, as shown in the descriptive statistics. Learning to code using ScratchJr is not the only approach where young children can develop their CT skills, after all. It is entirely possible that while the treatment group received the CAL curriculum, students in the control group were exposed to other STEM activities which were also helpful for developing their CT abilities. In addition, it is possible that the increase in both groups on TechCheck from pre to post-test was due to a test-retest effect.

Future research with larger sample sizes and higher fidelity of implementation is warranted to replicate the findings and provide more evidence regarding the impact of the CAL curriculum on students' CT skills. Study designs where the control group receives a non-CS related curriculum are also encouraged to better understand the efficacy of the current curriculum.

The interaction between student demographic characteristics and intervention outcomes

Research has found that students from low SES backgrounds and those who are English learners are usually disadvantaged in CS programs due to reasons such as lack of resources, teacher stereotypes, and language issues (Jacob et al., 2018; Santo et al., 2019; Su et al., 2023). However, based on the findings of the current study, while receiving CAL instruction, students from low SES and LEP backgrounds gained equal achievements in their coding skills compared with their peers. These findings could be explained by the following reasons.

First, evidence has shown that students from low SES backgrounds are more likely to have lower cognitive and academic skills (Dotson et al., 2009; White, 1982). However, the ScratchJr programming language is low-floor-high-ceiling, enabling students with various cognitive, mathematical, and analytical skills to engage, learn, and grow (Flannery et al., 2013). This means that the CAL curriculum created equal opportunities and space for improvement for all students, regardless of their baseline cognitive and academic skills and family socioeconomic status.

Second, previous studies have emphasized that one of the major challenges for English learners in learning to code is that most programming languages are text-based with strict syntactic rules, which can be unintuitive for emerging readers, especially language learners, and can add more frustration during debugging. In contrast to most other programming languages on the market, ScratchJr is a block-based programming language that is much more intuitive and self-explanatory. With the different shapes, symbols, and colors of the blocks, syntactic demands are lowered. These factors consequently decrease frustration and confusion while increasing the engagement and motivation for the learners, particularly those who may struggle with articulating themselves in the dominant language in the classroom during troubleshooting (Jacob et al., 2018).

These findings are encouraging because they provide critical pieces of evidence that the CAL curriculum is promising in providing equal opportunities for CS education for English learners and students from low SES backgrounds. The finding that students with low SES tended to develop more than their more affluent peers in CT skills sheds light on the potential of the CAL curriculum to close the achievement gap in CS education for students from various social classes.

It is also worth noting a borderline effect where students from low SES backgrounds may have achieved greater growth in CT skills than their more affluent peers over the CAL implementation. This may be because students from less affluent families tended to lack the necessary exposure to and experiences with technology and related STEM education resources that are key to developing CT skills (Grover et al., 2016). During the intervention, the CAL curriculum introduced some key CT concepts such as abstraction and modularity, which might be brand-new for students with low SES backgrounds but already familiar concepts for students from affluent backgrounds. Given that this finding is a borderline effect, no conclusions can be drawn at this point, and future research should be conducted to replicate the study with larger samples to provide more solid evidence with greater statistical power to guide educational practice, research, and policy.

In addition, we found that although students with disabilities and female students achieved similarly in CT, they had smaller gains in coding compared with their peers. The

CAL curriculum was designed to break gender stereotypes in CS fields by making the curriculum culturally responsive and introducing stories about female coders and women in CS and STEM, such as the inclusion of Grace Hopper: Queen of Computer Code in the second-grade curriculum.

One possible explanation for the finding on the interactions with gender and disability is teacher belief and bias. Prior studies have pointed out that teachers often have lower expectations for girls in STEM (Newall et al., 2018) and for students with disabilities in nearly all subjects (Rousso, 2003). The CAL curriculum includes activities where students create their own projects and teachers walk around the class to help and assist. Lower expectations may lead teachers to spend less time with certain students or may set up goals that are easier to reach for certain students. The lack of individualized instruction, along with less exposure and practice with more advanced blocks, may have contributed to the smaller growth observed.

Another possible reason that might have led to the interaction with students' disability status is that ScratchJr may be more difficult for students with certain types of disabilities to access, such as blindness. The limited access to assistive technologies and software may have created additional difficulties for this student population to engage in class and with the curriculum.

The finding on the gender difference is consistent with the majority of studies in this line of research, which suggest that boys tend to outperform girls in STEM learning (see a review study by Su et al., 2023). However, it contradicts a recent experiment on robotics programming programs for preschoolers, where researchers did not find any gender differences in computational thinking outcomes (Yang et al., 2023). The current study uses ScratchJr as the programming language, which is screen-based, while the study by Yang and colleagues used Matatalab, a screen-free coding robot kit. It is possible that girls and boys responded differently to the various coding instructions, activities, and learning dynamics between these two programs.

These findings highlight the importance of further research to explore the impact of gender on children's STEM learning and teaching in different contexts using different educational tools. Qualitative research that explores how female students and students with disabilities interact with teachers, compared to their peers, during the curriculum implementation may also yield important insights. More importantly, more effort and attention should be directed towards the improvement of current CS curriculums and the development of future CS curriculums that promote inclusive excellence.

Limitations

Several limitations must be considered while interpreting the aforementioned findings. First, the COVID-19 pandemic presented unexpected challenges during the study's implementation. For instance, the pre- and post-assessments had to be conducted remotely, and students at site B (20%) were unable to complete all 24 lessons before the post-assessments due to administrative reasons. Second, due to safety and health concerns at the schools, researchers were not allowed to go to the classrooms for observations during the study. As a result, it was challenging for the researchers to gather detailed data on how the CAL curriculum was implemented in the treatment schools, the regular curriculum used by the schools in the control group, or monitor the intervention's validity in day-to-day instructions. However, to ensure implementation fidelity, all participating teachers received professional development training and completed the CSA assessment before the program started. Moreover, other sources of data were collected, including the lesson logs from the teachers and teacher interviews after the study.

Conclusion

The current study conducted a randomized controlled trial to examine the efficacy of a CS curriculum CAL on the programming and CT skills for students in kindergarten, first, and second-grade classrooms. Our findings revealed that the CAL curriculum was effective in improving children's programming skills using the ScratchJr coding language, and a larger effect was found for first and second graders compared to kindergarteners. However, we did not find a statistically significant difference in students' CT skills between the treatment and the control groups. By examining the interaction between students' demographic characteristics and the intervention condition, we found that students from disadvantaged groups, such as students with disabilities or female students, did not learn as much as their peers. In addition, students with limited English proficiency and those from low SES backgrounds achieved similar gains in coding as their peers. These findings shed light on the potential of the CAL curriculum as an effective and developmentally appropriate CS curriculum for children in kindergarten to second-grade classrooms in improving their programming skills. The results also provide concrete ideas for future studies, the results of which could contribute to the development of more inclusive STEM curricula and instructions that strive for inclusive excellence for all students.

Acknowledgments

We would like to express our appreciation to the students and teachers who participated in the study, as well as the research assistants and researchers at the DevTech Research group. We would like to also thank the Shaffer Evaluation Group for supervising and monitoring the study design, data collection and evaluation. Additionally, we extend our sincere gratitude to the Scratch Foundation for their invaluable support.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Department of Education IER program grant # U411C190006.

ORCID

Dandan Yang http://orcid.org/0000-0001-8175-6931

References

Baber, L. D. (2015). Considering the interest-convergence dilemma in STEM education. *The Review of Higher Education*, *38*(2), 251–270. https://doi.org/10.1353/rhe.2015.0004

Bell, T., & Lodi, M. (2019). Constructing computational thinking without using computers. *Constructivist Foundations*, *14*(3), 342–351.

Bers, M. (2006). The role of new technologies to foster positive youth development. *Applied Developmental Science*, *10*(4), 200–219. https://doi.org/10.1207/s1532480xads1004_4

Bers, M. U. (2012). *Designing digital experiences for positive youth development: From playpen to playground*. Oxford.

Bers, M. U. (2018). Coding and computational thinking in early childhood: The impact of ScratchJr in Europe. *European Journal of STEM Education*, *3*(3), 8. https://doi.org/10.20897/ejsteme/3868

Bers, M. U. (2022). *Beyond coding: How children learn Human values through programming*. The MIT Press.

Bers, M. U., Blake-West, J., Kapoor, M. G., Levinson, T., Relkin, E., Unahalekhaka, A., & Yang, Z. (2023). Coding as another language: Research-based curriculum for early childhood computer science. *Early Childhood Research Quarterly*, *64*, 394–404. https://doi.org/10.1016/j.ecresq.2023.05.002

Bers, M. U., & Sullivan, A. (2019). Computer science education in early childhood: The case of ScratchJr. *Journal of Information Technology Education: Innovations in Practice*, *18*, 113–138. https://doi.org/10.28945/4437

Betancur, L., Votruba-Drzal, E., & Schunn, C. (2018). Socioeconomic gaps in science achievement. *International Journal of STEM Education*, *5*(1), 1–25. https://doi.org/10.1186/s40594-018-0132-5

Blums, A., Belsky, J., Grimm, K., & Chen, Z. (2017). Building links between early socioeconomic status, cognitive ability, and math and science achievement. *Journal of Cognition and Development*, *18*(1), 16–40. https://doi.org/10.1080/15248372.2016.1228652

Bouck, E. C., & Yadav, A. (2022). Providing access and opportunity for computational thinking and computer science to support mathematics for students with disabilities. *Journal of Special Education Technology*, *37*(1), 151–160. https://doi.org/10.1177/0162643420978564

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association*, Vancouver, Canada (Vol. 1, p. 25).

Buxton, C. A., & Lee, O. (2014). English learners in science education. In D. Treagust, & C.-Y. Cui (Eds.), *Handbook of research on science education, volume II* (pp. 218–236). Routledge.

Campbell, F. A., Pungello, E. P., Miller-Johnson, S., Burchinal, M., & Ramey, C. T. (2001). The development of cognitive and academic abilities: Growth curves from an early childhood educational experiment. *Developmental Psychology*, *37*(2), 231. https://doi.org/10.1037/0012-1649.37.2.231

Chachashvili-Bolotin, S., Milner-Bolotin, M., & Lissitsa, S. (2016). Examination of factors predicting secondary students' interest in tertiary STEM education. *International Journal of Science Education*, *38*(3), 366–390. https://doi.org/10.1080/09500693.2016.1143137

Cheryan, S., Master, A., & Meltzoff, A. N. (2015). Cultural stereotypes as gatekeepers: Increasing girlsâ €™ interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology*, *6*, 49. https://doi.org/10.3389/fpsyg.2015.00049

Cobb, P., Confrey, J., DiSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, *32*(1), 9–13. https://doi.org/10.3102/0013189X032001009

Code.org Advocacy Coalition. (2017). State policies on K-12 computer science: An overview. Retrieved from https://www.code.org/advocacy/state-policy/overview

Computer Science for All. (2016). Retrieved from https://obamawhitehouse.archives.gov/computer-science-for-all

De Ruiter, L. E., & Bers, M. U. (2021). The coding stages assessment: Development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language. *Computer Science Education*, *32*(4), 388–417. https://doi.org/10.1080/08993408.2021.1956216

Dotson, V. M., Kitner-Triolo, M. H., Evans, M. K., & Zonderman, A. B. (2009). Effects of race and socioeconomic status on the relative influence of education and literacy on cognitive functioning. *Journal of the International Neuropsychological Society*, *15*(4), 580–589. https://doi.org/10.1017/S1355617709090821

Fayer, S., Lacey, A., & Watson, A. (2017). STEM occupations: Past, present, and future. *Spotlight on Statistics*, *1*, 1–35. https://stats.bls.gov/spotlight/2017/science-technology-engineering-and-mathematics-stem-occupations-past-present-and-future/pdf/science-technology-engineering-and-mathematics-stem-occupations-past-present-and-future.pdf

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013, June). Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th international conference on interaction design and children*, New York, NY, USA (pp. 1–10).

Friedman-Krauss, A. H., Barnett, W. S., Garver, K. A., Hodges, K. S., Weisenfeld, G. G., & Gardiner, B. A. (2021). *The state of preschool 2020: State preschool yearbook*. National Institute for Early Education Research.

Garces, E., Thomas, D., & Currie, J. (2002). Longer-term effects of head start. *American Economic Review*, *92*(4), 999–1012. https://doi.org/10.1257/00028280260344560

García Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers. TACCLE3 Consortium, Belgium. https://doi.org/10.5281/zenodo.165123

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

Hwang, J., & Taylor, J. C. (2016). Stemming on STEM: A STEM education framework for students with disabilities. *Journal of Science Education for Students with Disabilities*, *19*(1), 39–49. https://doi.org/10.14448/jsesd.09.0003

Israel, M., Wherfel, Q. M., Pearson, J., Shehab, S., & Tapia, T. (2015). Empowering K–12 students with disabilities to learn computational thinking and computer programming. *Teaching Exceptional Children*, *48*(1), 45–53. https://doi.org/10.1177/0040059915594790

Jacob, S. R., Montoya, J., Nguyen, H., Richardson, D., & Warschauer, M. (2022). Examining the what, why, and how of multilingual student identity development in computer science. *ACM Transactions on Computing Education (TOCE)*, *22*(3), 1–33. https://doi.org/10.1145/3500918

Jacob, S., Nguyen, H., Tofel-Grehl, C., Richardson, D., & Warschauer, M. (2018). Teaching computational thinking to English learners. *NYS Tesol Journal*, *5*(2), 12–24.

Jiang, S., Simpkins, S. D., & Eccles, J. S. (2020). Individuals' math and science motivation and their subsequent STEM choices and achievement in high school and college: A longitudinal study of gender and college generation status differences. *Developmental Psychology*, *56*(11), 2137. https://doi.org/10.1037/dev0001110

K–12 computer science framework. (2016). Retrieved from http://www.k12cs.org.

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

Kapoor, M. G., Yang, Z., & Bers, M. (2023). Supporting early elementary teachers' coding knowledge and self-efficacy through virtual professional development. *Journal of Technology & Teacher Education*, *30*(4), 1–31.

Klimaitis, C. C., & Mullen, C. A. (2021). Access and Barriers to Science, Technology, Engineering, and Mathematics (STEM) Education for K–12 Students with Disabilities and Females. In C. Mullen (Ed.), *Handbook of Social Justice Interventions in Education* (pp. 813–836). Springer.

McGee, E. O. (2021). *Black, brown, bruised: How racialized STEM education stifles innovation*. Harvard Education Press.

Milner-Bolotin, M., & Marotto, C. C. (2018). Parental engagement in children's STEM education. Part I: Meta-analysis of the literature. *LUMAT: International Journal on Math, Science and Technology Education*, *6*(1), 41–59. https://doi.org/10.31129/LUMAT.6.1.292

National Science Board. (2018). Expanding access to computer science education in K-12 schools. Retrieved from https://www.nsf.gov/nsb/policy/cse/expanding-access-computer-science-education-k-12-schools

Newall, C., Gonsalkorale, K., Walker, E., Forbes, G. A., Highfield, K., & Sweller, N. (2018). Science education: Adult biases because of the child's gender and gender stereotypicality. *Contemporary Educational Psychology*, *55*, 30–41. https://doi.org/10.1016/j.cedpsych.2018.08.003

Papert, S. (1980). *Children, computers and powerful ideas* (Vol. 10). Basic Books.

Portelance, D. J., Strawhacker, A., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, *26*(4), 1–16. https://doi.org/10.1007/s10798-015-9325-0

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

Rousso, H. (2003). Gender and Education for All: The Leap to Equality. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2ab4120ba42d64cca276e11785d92740ba18d781

Ruhm, C., & Waldfogel, J. (2012). Long-term effects of early childhood care and education. *Nordic Economic Policy Review*, *1*(1), 23–51.

Santo, R., DeLyser, L. A., Ahn, J., Pellicone, A., Aguiar, J., & Wortel-London, S. (2019, February). Equity in the who, how and what of computer science education: K12 school district conceptualizations of equity in 'cs for all initiatives. In *2019 research on equity and sustained participation in engineering, computing, and technology (RESPECT)*, Minneapolis, MN, USA (pp. 1–8). IEEE. https://doi.org/10.1109/RESPECT46404.2019.8985901

Sarama, J., & Clements, D. H. (2009). *Early childhood mathematics education research: Learning trajectories for young children*. Routledge.

Stamatios, P. (2022). Can preschoolers learn computational thinking and coding skills with ScratchJr? A systematic literature review. *International Journal of Educational Reform*, *10567879221076077*, 105678792210760. https://doi.org/10.1177/10567879221076077

Su, J., Yang, W., & Zhong, Y. (2023). Influences of gender and socioeconomic status on children's use of robotics in early childhood education: A systematic review. *Early Education and Development*, *34*(4), 910–926. https://doi.org/10.1080/10409289.2022.2078617

Thomas, J., Utley, J., Hong, S. Y., Korkmaz, H., & Nugent, G. (2020). Parent Involvement and Its Influence on Children's STEM Learning: A review of the Research. In C. C. Johnson, L. D. English, M. J. Mohr-Schroeder, & T. Moore (Eds.), *Handbook of Research on STEM Education* (p. 26). Taylor & Francis.

Unahalekhaka, A., & Bers, M. U. (2021). Taking coding home: Analysis of ScratchJr usage in home and school settings. *Educational Technology Research & Development*, *69*(3), 1579–1598. https://doi.org/10.1007/s1RR1423-021-10011-w

White, K. R. (1982). The relation between socioeconomic status and academic achievement. *Psychological Bulletin*, *91*(3), 461. https://doi.org/10.1037/0033-2909.91.3.461

Wing, J. M. (2006). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *364*(1844), 1171–1178.

Wing, J. M. (2011). Computational thinking: What and why? *Communications of the ACM*, *54*(3), 66–69.

Wurman, Z. E., & Donovan, W. (2020). Breaking the code: The state of computer science education in America's public schools. White Paper No. 206. *Pioneer Institute for Public Policy Research*.

Xie, Y., Fang, M., & Shauman, K. (2015). STEM education. *Annual Review of Sociology*, *41*(1), 331. https://doi.org/10.1146/annurev-soc-071312-145659

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