

A Cross-Sectional Study Investigating Primary School Children's Coding Practices and Computational Thinking Using ScratchJr

Journal of Educational Computing
Research
0(0) 1–38

© The Author(s) 2021





Article reuse guidelines:

sagepub.com/journals-permissions

DOI: 10.1177/07356331211027387

journals.sagepub.com/home/jec



Eleni A. Kyza , Yiannis Georgiou ,
Andria Agesilaou , and
Markos Souropetsis 

Abstract

There are increasing calls to introduce computational thinking in schools; the arguments in favor call upon research suggesting that even kindergarten children can successfully engage in coding. This contribution presents a cross-sectional study examining the coding practices and computational thinking of fifty-one primary school children using the ScratchJr software; children were organized in two cohorts (Cohort 1: 6–9 years old; Cohort 2: 10–12 years old). Each cohort participated in a six-hour intervention, as part of a four-day summer club. During the intervention children were introduced to ScratchJr and were asked to collaboratively design a digital story about environmental waste management actions, thus adopting a disciplinary perspective to computational thinking. Data analyses examined children's final artifacts, in terms of coding practices and the level of computational thinking demonstrated by each cohort. Furthermore, analysis of selected groups' storyboard interviews was used to shed light on differences between the two cohorts.

Media, Cognition and Learning Research Group, Department of Communication and Internet Studies, Cyprus University of Technology, Limassol, Cyprus

Corresponding Author:

Eleni A. Kyza, Department of Communication and Internet Studies, Cyprus University of Technology, P.O. Box 50329, Limassol 3603 Cyprus.

Email: Eleni.Kyza@cut.ac.cy

Results are presented and contrasted across the two age cohorts via a developmental perspective. The findings of this study can be useful in considering the instructional support that is necessary to scaffold the development of primary school children's coding practices and computational thinking.

Keywords

coding practices, computational thinking, ScratchJr, primary school children, cross-sectional design

Computational Thinking (CT) has been recognized as a set of competencies that can promote analytical thinking, which is often linked to economic growth (Horizon K-12 report, 2016). The emphasis on CT has been spurred by seminal calls such as the ones put forth by Seymour Papert and Jeannette Wing, who highlighted the potential of CT to support analytical thinking, creativity and the development of critical thinking skills. Recently, there has been an increased emphasis on CT around the world, including discussions about integrating programming and CT in primary education curricula (Kong & Wang, 2021; Sun et al., 2021). At the same time, numerous technological products (e.g., software and robotic kits) have emerged aiming at supporting children to learn how to code (Jung & Won, 2018; Papadakis, 2020; Sullivan & Heffernan, 2016; Yu & Roque, 2019), and for developing the CT competencies needed in an ever-increasing technological world. The introduction of coding and CT to K-12 education is being supported by “low threshold, high ceiling” software, such as ScratchJr (Flannery et al., 2013). Research using this software to-date has provided initial empirical evidence that coding can be learned by children as young as four years old (Bers, 2012), and can promote children's general and higher order thinking skills (Fallon, 2016).

The study described in this paper focuses on primary school children's coding practices and understanding of underlying CT concepts using Scratch Jr. It is motivated by the small number of empirical studies on the topic, as the majority of empirical research on CT has been conducted at the higher education level (Heintz et al., 2016). According to Nouri et al. (2020) programming and CT in K-9 education are relatively new and available research is still very limited. However, due to the existence of new programming languages that are developmentally appropriate for young children, there is a need to extend research, as it is not sufficient to assume that research findings with college students can be

transferred to younger populations (Grover & Pea, 2013; Lye & Koh, 2014; Nouri et al., 2020).

One of the areas that has been flagged as needing more research is the developmental trajectory of CT. According to Bers (2019) “describing the activity of coding as a learning progression assumes a developmental approach supported by instruction that takes into account both cognitive as well as socioemotional factors” (p. 505). Despite the fact that there is extensive work on defining learning progressions in various other disciplines, such as childhood literacy or mathematics education (Bers, 2019), limited work has been conducted so far regarding CT with younger children. Such research gaps emphasize the need for studies reporting on young children’s efforts to code, to inform research and practice regarding children’s learning progressions on the development of CT.

Toward this end, this study reports on primary school children’s coding practices and CT, adopting a developmental perspective via a cross-sectional design (Cummings, 2018). During a six-hour intervention, children were introduced to ScratchJr to collaboratively design a digital story about environmental waste management. Children’s artifacts were analyzed to examine their coding practices and their relation to CT, thus providing useful insights about each cohort. The study methodology is aligned with an increasing corpus of research which reports that such cross-sectional studies on children’s understanding and skills at different developmental stages and ages can contribute to the development of hypothetical progressions for instructional pathways and curriculum planning (e.g., Cadorna et al., 2021; Duncan et al., 2009; Duschl, 2019).

Theoretical Framework

The Value of Teaching Coding and CT

The teaching of coding in K-12 education and the promotion of CT practices have gained ground in recent years (Barcelos et al., 2018; Buitrago Flórez et al., 2017; Govind et al., 2020; Lye & Koh, 2014). This growing trend has been justified from multiple perspectives. As argued, through coding young programmers can reinforce various cognitive skills, such as literacy, number sense, critical thinking and creativity, necessary to succeed in today’s digital world (Clements, 1999; Jakoš & Verber, 2017; Tran, 2019; Wing, 2006). In addition, via the teaching of coding, primary school children could obtain a better understanding of concepts grounded not only in the field of computer science, but also across the curriculum (Baytak & Land, 2011; Brennan et al., 2011; Shodiev, 2015). Researchers have also supported that, when coding, children could employ a range of new media to communicate and test their ideas and designs (Portelance et al., 2016; Resnick, 2013). Overall, it seems that coding has emerged as a new form of literacy (Papadakis et al., 2016). This position indicates that, similar to reading and writing, CT may facilitate children to develop substantial skills, such as the

development of algorithmic, problem-solving and computational competencies (Fessakis et al., 2013; Kafai & Burke, 2014).

Even though there are different definitions of CT (Grover & Pea, 2013), most agree that CT is centered around solving problems, requires the application of analytic and logical thinking about data to solve a problem, and that developing CT is not necessarily or always dependent on using computers. One of the most cited definitions has been provided by Jeannette Wing (2006), who explained CT as “the thought processes involved in formulating a problem and expressing its solution in a way that a computer—human or machine—can effectively carry out” (p. 7). A definition provided by the United States National Research Council (NRC) further indicates that CT is “the process of recognising aspects of computation in the world that surrounds us and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes” (National Research Council, 2011, p. 29).

The emerging nature of CT still leaves many open questions on the best ways to conceptualize it and to assess it, especially when it comes to younger people (Hsu et al., 2018). Brennan and Resnick (2012) have described a framework for examining young people’s CT that takes into account CT *concepts* (such as sequences, loops, and events), *design practices* (e.g., being incremental and iterative, testing and debugging, etc.), as well as the *learners’ perspectives* and how these perspectives change over time. Moreno-León et al. (2015) offered an operationalization of assessing students’ CT concepts in Scratch projects, based on a synthesis of the literature. This operationalization identified seven CT concepts: (1) abstraction and problem decomposition, (2) logical thinking, (3) synchronization, (4) parallelism, (5) algorithmic notions of flow control, (6) user interactivity and (7) data representation.

Despite the complex nature of CT, the teaching of coding is nowadays considered acceptable even for pre-school children. Several countries, such as the United Kingdom, have reformed their educational curricula to include coding lessons for children as young as five (Papadakis et al., 2016). An increasing number of organizations and institutions, such as Code.org and the Code-to-Learn Foundation, have been offering opportunities for young children to learn how to code (Portelance et al., 2016). Various visual-based programming tools have emerged aiming at supporting young children to develop coding practices. The following section provides more information on the role of visual-based programming to support children’s coding and the development of CT.

The Role of Visual-Based Programming in Addressing Children’s Difficulties With Coding and CT

Despite the value of coding and CT, it is reported that children face several challenges as programming novices. Very young children have not started, or are just beginning, to read and have fewer linguistic experiences than adults.

Children may also have difficulties with typing and may not have the motor skills required to control a mouse, especially for longer periods of time (Clarke-Midura et al., 2019; Pila et al., 2019). At the same time, young children lack logical reasoning, while their still developing critical thinking skills (Robins et al., 2003) may result in poor understanding of the rules, logic, and syntax of programming languages (Ala-Mutka, 2004). In general, children may engage in “magical” thinking rather than engaging in reasoning to understand the actions of machines or the algorithmic thinking needed to understand how programs are executed (Flavell et al., 1993; Mioduser et al., 2009; Pea, 1986).

The literature also indicates several difficulties and misconceptions that young children, as novice programming learners, encounter (Baser, 2013; Durak, 2016; Fokides, 2018; Žanko et al., 2019). For instance, beginning programmers may face challenges in developing simple algorithms (Kraleva et al., 2019), in understanding and using abstract representations such as variables (Fields et al., 2015; Hermans & Aivaloglou, 2017; Relkin et al., 2021), in grasping “if-then” conditionals (Barrouillet & Lecas, 1999; Müller et al., 2001), or in detecting and handling errors in their programs (Clarke-Midura et al., 2019).

Using simple and age-appropriate programming environments to mitigate the various difficulties in learning programming is of crucial significance for young students (Mladenovic et al., 2018). Toward this end, various block-based programming tools have been developed to support novice learners in learning how to code while also contributing to the development of their CT. With these programming tools the coding process can be performed via drag and drop of visual objects, known as blocks, using a modular editing interface which allows learners to develop and execute programs in a more intuitive and user-friendly way (Chao, 2016; Durak, 2020; Hu et al., 2021; Lye & Koh, 2014). This drag and drop process keeps young children more focused on the programming process itself and decreases their cognitive load (Durak, 2020; Mladenovic et al., 2018; Moors et al., 2018; Vasilopoulos & Van Schaik, 2019). In contrast to text-based programming, visual programming does not require children memorizing complex programming syntax or debugging errors in their codes (Durak, 2020; Lindberg et al., 2019). Instead, these programming tools are argued to enable the introduction of children to coding, via facilitating their understanding of how to use basic algorithmic forms through a smooth and straightforward visual mode of programming (Portelance & Bers, 2015).

Such visual-based environments enable children to construct meaningful programs with instant output, which in turn sustains their engagement with the programming process (Bers, 2018; Kong & Wang, 2019). Active engagement is significant in learning programming, otherwise, students’ interest, artifacts and learning can be affected (Durak, 2020). Visual programming tools transform young children into active creators of games, animations, and interactive stories in various disciplines (e.g., language, science, or mathematics), which can boost their interest in CT learning (Mladenovic et al., 2018; Moors et al., 2018). Many

of these visual-based tools work on touch devices (tablets) whose interactive interface makes them even more accessible to young learners and children (Hill et al., 2015).

The literature suggests that visual programming tools are most often used to foster the acquisition of coding skills and CT in educational contexts (de Araujo et al., 2016; Resnick et al., 2009; Shute et al., 2017), as they effectively facilitate the cognitive, motor, and social development of young children (Lee et al., 2013). This study examines CT using ScratchJr, which was especially designed for younger children and, at the time of this study, only worked on tablets. We briefly present ScratchJr in the next section.

The ScratchJr Programming Language

ScratchJr was designed for early childhood students (ages 5–7) and seeks to support them in understanding basic programming concepts in a developmentally appropriate manner (Flannery et al., 2013). It is an introductory programming environment that was designed for introducing young children to coding and CT via allowing them to create projects in the form of interactive stories, collages and games, which cater to children’s cognitive, personal and emotional development (Flannery et al., 2013; Papadakis et al., 2016; Portelance et al., 2016). ScratchJr provides a visual environment, consisting of “programming blocks”; sequences of blocks can be connected to control the sprites (characters or objects) that appear on the tablet’s screen (Portelance et al., 2016). The programming blocks have the form “of jigsaw puzzles pieces with visual properties that correspond to their syntactic properties” (Portelance et al., 2016, p. 491). Many of the more advanced features of Scratch (e.g., variables), have been removed to create a more age-appropriately programming environment for younger children (Papadakis et al., 2016).

The ScratchJr programming language includes a total of 28 blocks, which are grouped by color and represent six functions (ScratchJr.org, 2015a, 2015b): (a) yellow “Trigger” blocks (placed at the start of a script to make that script execute when a certain action happens), (b) blue “Motion” blocks (to make characters move), (c) purple “Looks” blocks (to change how characters look), (d) green “Sound” blocks (to play a sound from ScratchJr’s library), (e) orange “Control flow” blocks (to change the nature of a character’s program), and (f) red “End” blocks (placed at the end of the script to define when the program finishes executing). Children can drag and snap together these blocks for creating scripts, to control the sprites’ motions, appearance, and interactions. Projects can be developed by adding up to four pages, which can integrate sprites (characters and objects) and/or backgrounds. The ScratchJr interface is composed of four different sections, as follows: (a) The programming area, where the children drag and connect their blocks to create scripts; (b) The stage, in which the sprites (characters and objects) appear, move and interact

according to the scripts; (c) List of characters, where all characters integrated in the stage appear; and, (d) Pages, where all the project pages appear. Each page is associated with a new scene allowing the continuation of the project.

Investigating Children's Coding and CT via ScratchJr

The affordances of ScratchJr, and the fact that there are limited programming tools for pre-primary and early primary school children, have made it popular around the globe; nonetheless, empirical research focusing on children's coding practices via ScratchJr as well as its impact on children's CT is still in its infancy. Existing studies have largely focused on how ScratchJr has been adopted (e.g., Flannery et al., 2013; Strawhacker et al., 2015).

Portelance et al. (2016) investigated K-2 children's block-based choices when programming via ScratchJr. While motion-based blocks were embraced by the children across the three grades studied, there were notable differences in children's block usage between the three grades. For instance, second graders used significantly fewer "End" and "Trigger" blocks than kindergartners. The researchers attributed this finding to the lack of a direct impact on the sequence, as the older children may have discovered that their programming sequences were still functional without the use of these blocks. According to the researchers, another plausible explanation was related to the symmetry these blocks provide in the sequence, which may be more captivating for younger children. In contrast, second graders used the "Control" and "Sound" blocks more than children in lower grades, as these blocks may have been more complex for younger children.

Papadakis et al. (2016) conducted an exploratory case study with 43 preschool children using ScratchJr during a 13-hour intervention and found that: (a) there were no statistically significant differences in performance in computational and digital skills between boys and girls, (b) the most used blocks were the "Motion" blocks, and (c) preschoolers had more difficulties when the scene included the coding of more than one character.

In a more recent study, Strawhacker and Bers (2019) investigated K-2 children's performance on a programming assessment after engaging in a 6-week curricular intervention. Results showed that while the participating children mastered foundational coding concepts, there were significant differences in children's performance and subsequent understanding across the three grade levels. For instance, the higher a child's grade in school, the higher his/her performance on ScratchJr block recognition. They also reported that while most of the children mastered simple "Motion" commands, children at the kindergarten level had more difficulties with meta-level "Control flow" blocks, which do not program characters directly but instead modify how programs are executed. Likewise, children at the kindergarten level had more trouble in coordinating multiple characters.

Overall, empirical research investigating children's programming with ScratchJr appears to have mainly focused on the coding practices manifested by the children, rather than on the computational concepts underlying these practices. We have identified only a very recent study by Chou (2020) which has investigated the development of young children's CT. As part of this study, Chou (2020) provided empirical evidence suggesting that third graders immersed in weekly programming projects using ScratchJr significantly improved their CT concepts (sequence, event, and parallelism) and practices (testing and debugging, as well as reusing and remixing). However, according to Chou (2020) a limitation was that the study investigated children's programming learning progress during a CT project and did not analyze children's CT using their programming outcomes (artifacts). In addition, the study only adopted a one-group pretest and posttest design with only third graders as participants. In this context, Chou has suggested that future studies should employ comparison groups with children of different ages, as age may influence cognitive development during programming training as well as for further exploring CT and coding patterns based on age.

At the moment, existing studies are limited to K-3 age children, as this is the official target group of ScratchJr and, and as older children can use the more advanced Scratch version. However, some researchers have argued that Scratch may be too advanced for children in grades 4–6 (e.g., Hill et al., 2015). At the same time, as argued by Papadakis et al. (2016), even though ScratchJr was designed with younger children (K-3) in mind, this does not limit the potential complexity of the designs. Based on these remarks it is safe to conclude that ScratchJr may provide a K-6 programming environment that can successfully introduce children of 4–12 years-old to coding and CT.

Rationale and Research Questions

In this study, we sought to investigate the coding practices, but also, the underlying CT concepts between two cohorts of children in primary school, ages 6–12 years old (Cohort 1: Lower primary education children, Cohort 2: Upper primary education children) using ScratchJr. According to Sáez-López et al. (2016) only a few empirical studies have focused on the investigation of computing in primary school settings, while such cross-sectional comparisons with primary school children have been reported only for K-3 children's coding practices (e.g., Chou, 2020; Portelance et al., 2016). At the same time, researchers have suggested the need for artifact-based content analysis approach to investigate children's CT (Chou, 2020). In this context, we pursued the following four research questions:

1. What are the main differences in children's coding practices in terms of the programming blocks used between the chosen age cohorts? (RQ1)

2. What are the main differences regarding the quality and complexity of coding, between the two cohorts? (RQ2)
3. What are the main differences in the level of CT demonstrated by each cohort? (RQ3)
4. How do the children in each cohort apply CT concepts to create a digital story about a waste management issue? (RQ4)

To answer these questions, we examined differences and similarities between the two cohorts' coding practices and underlying CT concepts, as these were exhibited in the ScratchJr projects they co-authored.

Methods

This study employed a mixed-methods design, which can offer depth and breadth in understanding the data and allows for triangulating the findings (Creswell & Clark, 2007). As argued by Portelance (2015) "triangulating student artifacts like programming projects with other data about student learning can more thoroughly evaluate learning trajectories and outcomes" (p. 8). In the following sections, we provide an overview of the methodology of the present study.

Participants

Fifty-one (51) primary school children participated in this study. This was a convenience sample, as these children were enrolled in a summer school organized at the premises of a public university in Cyprus. Children's participation in the educational intervention was voluntary and all necessary permissions and informed consents, including those of the children's guardians, were received prior to the study. Permission was provided by the university authorities to conduct this research study during the summer club at the university premises. None of the children had previously used the ScratchJr software nor had extended prior experiences in coding, as programming had not yet been integrated in the national, centralized curriculum of Cyprus. This finding was also corroborated through the children's post-activity interviews. While some of the children reported that they had previously participated in educational activities aiming at storytelling, none of the children had previously used ScratchJr or any other programming tool. Three of the four authors were present in all sessions, taking responsibility for leading the two cohorts. No other teacher was present during the sessions.

The ScratchJr Curriculum and Learning Activities

Even though ScratchJr is primarily addressed to young children this does not limit the affordances of the programming environment which allow

diversification and complexity in the produced projects (Papadakis et al., 2016). As such, ScratchJr can be also used by older children in primary school, rather than Scratch which is a more demanding programming environment (Hill et al., 2015), while also providing ground for investigating children's computational practices across the K-6 spectrum.

The children were divided into two cohorts according to their age. These cohorts were selected because they also represent how primary education is organized by the national Ministry of Education:

- Cohort 1-Lower Primary Education (6–9 years old): Thirty-three children attending grades 1–3 (19 girls and 14 boys).
- Cohort 2-Upper Primary Education (10–12 years old): Eighteen children attending grades 4–6 (9 girls and 9 boys).

Children were subsequently assigned to groups of two or three according to their preference; as such, they ended up working in primarily same aged and same sex groups. Fifteen groups ($n = 15$) were formed for Cohort 1, and nine groups ($n = 9$) in Cohort 2.

Each group was asked to develop its own digital interactive story by coding in ScratchJr. The ScratchJr curriculum was structured around four 90-minute sessions for each cohort (one session a day for four consecutive days) and included the same content and pedagogical activities for the two cohorts. Children in both cohorts were provided with the same time for all learning activities (e.g., in session 2, the same time was allocated in both cohorts for mastering ScratchJr). Table 1 provides an overview of the educational intervention.

In Session 1, children were introduced to the concept of environmental waste management (3Rs: Reduce, Reuse, Recycle) via a sequence of experiential activities. In Session 2, children were taught how to use ScratchJr through a series of activities based on the “Animated Genres” curriculum, as described by Portelance and Bers (2015), and the activities included in ‘The Official ScratchJr Book’ (Bers & Resnick, 2015). Then, in Sessions 3 and 4 each group was provided with a tablet and children were asked to collaboratively develop a digital story using ScratchJr which could be shown to their families to motivate them to take responsible environmental waste management actions. To help children in their efforts, each group was first asked to work together to draw a storyboard, which they were then expected to transfer to ScratchJr. During the sessions, the instructors were rotating between groups to monitor the groups’ progress and provide additional scaffolding if needed. To promote ownership of the programming activity we asked children to switch who was handling the tablet after creating each project page.

Table 1. An Overview of the Educational Intervention.

Session	Learning Goals	Learning activities	Duration
1	Introduction to:	Introductory activity	10'
	• Problem-based situation	“Reduce” experiential activity	20'
	• Waste management (3Rs)	“Reuse” experiential activity	20'
	• ScratchJr	“Recycle” experiential activity	20'
		“Scratch Jr” free-exploration activity	20'
2	Introduction to:	Introductory activity	10'
	• Programming	Choosing background & sprites	20'
	• ScratchJr programming blocks	ScratchJr programming blocks	30'
	• Sequencing	Sequencing & repeating sequences	30'
3	Introduction to:	Introductory activity	15'
	• Storyboarding	Storyboard activity	45'
	• Digital storytelling	Digital storytelling activity	30'
4	Digital storytelling development	Digital storytelling activity	90'

Data Collection and Analysis

The data corpus included all group projects (see online appendix for an overview of the projects) to address RQ1, RQ2 and RQ3, as well as the groups’ storyboards, and individual interviews with the children regarding their storyboard design to address RQ4. In particular, to answer our research questions, we used artifact analysis techniques, which is an embedded assessment method. Embedded assessment is particularly effective for assessing younger children’s practices as it has greater ecological validity (Wilson & Sloane, 2000). According to Brennan and Resnick (2012) artifact-based interviews can provide, for instance, a more nuanced and personalized way of obtaining valuable insights regarding students’ coding practices and CT. The analysis of students’ artifacts (i.e., ScratchJr projects) was combined with story-based interviews, which are explained later in this paper. Each group’s ScratchJr project ($n = 24$) was collected, analyzed and assessed for (a) the type of programming blocks used, (b) the complexity and quality of coding (e.g., number of programmed sprites vs non-programmed sprites, functional scripts vs non-functional scripts), as well as (c) the level of CT according to six key CT concepts: *Abstraction and problem decomposition*; *Parallelism*; *Synchronization*; *Flow control*; *User interactivity*; and *Data representation*. Analyzing the block usage and their functionality, in relation to identifying the levels of children’s CT, can help to better understand children’s thought processes and behavioral practices in formulating the problem and suggesting a solution through programming.

The aforementioned comparisons between cohorts were analyzed descriptively and through statistical analysis with SPSS v.20.0 for Windows. In addition, we proceeded with the analysis of selected groups’ storyboard interviews (based

on the scoring of the projects), to shed light on the existence of any quantitative CT differences between the two cohorts.

Inter-rater reliability was established prior to scoring all projects. As part of this process, two projects from Cohort 1 (13.3% of the data corpus), as well as two projects from Cohort 2 (16.6% of the data corpus), were rated by two of the researchers independently. Cohen's κ was used to determine the agreement between the raters. There was high agreement for both inter-rater exercises: Cohort 1 projects ($\kappa = .907$, $p < .001$) and Cohort 2 projects ($\kappa = .811$, $p < .001$). All disagreements were discussed and resolved, with the researchers then individually assessing the remaining projects. In the next sections, we present the data analysis in more detail.

Programming Blocks. To address RQ1 (What are the main differences in children's coding practices in terms of the programming blocks used between the two cohorts?), the projects were first analyzed in terms of the different types of blocks employed (e.g., "Triggering" blocks, "Motion" blocks, "Looks" blocks) to identify coding trends within and across cohorts. Next, the programming block categories were compared between the cohorts and the average block usage by block category was calculated for each cohort. A Mann-Whitney U test was employed to investigate whether there were any statistical differences in the average block usage by block category between the two cohorts.

Quality and Complexity of Coding. To investigate RQ2 (What are the main differences regarding the quality and complexity of coding between the two cohorts?), the projects were analyzed to identify the quality and complexity within and between the two cohorts based on the pages, sprites and scripts that were created. Table 2 presents the main indicators per category (pages, sprites and scripts). For instance, when it came to scripts, we were not only interested in examining the number of scripts created but also in assessing their complexity. A Mann-Whitney U test was employed to investigate whether there were any statistically significant differences between the two cohorts, according to the three main aspects of ScratchJr (Pages, Sprites, Scripts) and their indicators.

Computational Thinking. We assessed CT at three levels (null, basic, developing). Table 3 shows a brief definition of each CT concept, which is based on Moreno-León et al.'s (2015) work, adapted for the age of the participants in this study and the affordances of the ScratchJr software they used.

To address RQ3 (What are the main differences in the level of CT demonstrated by each cohort?), projects were examined for evidence of CT, using the presence of the six key computational concepts (see Table 3) as indicators, and using three levels of coding practices for each one (Levels 0, 1, 2). Level 0 indicated the absence of the computational concept in children's coding as

Table 2. Assessing ScratchJr Projects in Terms of Pages, Sprites and Scripts.

Indicators	Definition
Pages	How many pages?
Backgrounds	How many backgrounds in the pages?
Unique backgrounds	How many unique backgrounds in the pages?
Sprites	How many sprites?
Programmed sprites	How many programmed sprites?
Non-programmed sprites	How many non-programmed sprites?
Default sprites	How many default sprites?
Edited sprites	How many edited sprites?
Scripts	How many scripts?
Functional scripts	How many functional scripts?
Total blocks	How many total blocks per script?
Blocks with numbers	How many blocks with numbers per script?

represented in their projects, Level 2 indicated a more sophisticated presence of the computational concept in the project pages, while Level 1 served as an intermediate stage. Each one of the six CT concepts was scored per project with 0, 1 or 2 marks respectively; the maximum score for a project was 12 points for all six CT concepts. In this way, for the 15 projects of Cohort 1 the maximum total score was 30 points per concept (15 projects X 2 points for each concept), while in the case of Cohort 2, the maximum total score was 18 (9 projects X 2 points for each concept). Inter-rater reliability was established using two projects from Cohort 1 and two projects from Cohort 2, which were independently rated by two researchers. Cohen's κ was used to determine the agreement between the raters. There was high inter-rater agreement ($\kappa=.862$, $p < .001$). After all disagreements were discussed and resolved, the researchers scored the remaining projects.

The assessment of the CT concepts in each project was followed by a descriptive analysis of (a) the percentage of projects per cohort exhibiting each level of CT, and (b) the percentage of the projects' total scores per CT concept in relation to the maximum score that the children could reach for each of the two cohorts. Next, the average overall score and score per computational concept category were calculated for each cohort and a Mann-Whitney U test was employed to investigate whether there were any statistical differences between the two cohorts.

Individual Storyboard Interviews. Each group's storyboard was collected, and short individual interviews were held with each member of each group to shed light on RQ4 (How do the children in each cohort apply CT concepts to create a digital story on a waste management issue?). In these interviews we sought to

Table 3. Rating Scheme for the Core Computational Thinking (CT) Concepts.

CT concept and definition	Levels of coding practices		
	Level 0 Null	Level 1 Basic (1 point)	Level 2 Developing (2 points)
<i>Abstraction and problem decomposition</i> Breaking down a complex problem into smaller parts, reducing complexity	One script, or one sprite, or none	More than one script, and more than one sprite in one of the project's pages	More than one script, and more than one sprite in all of the project's pages
<i>Parallelism</i> Making things happen at the same time	No script or only one script on green flag.	Two or more scripts on Green flag, but not all of them running correctly.	Two or more scripts on Green flag, and all of them running correctly.
<i>Synchronization</i> Synchronize events between sprites	Absence of synchronization.	Use of only one Wait "Control" block in combination with any of the "Looks", "Motion", and "Sound" blocks.	Use of the Wait "Control" block in combination with any of the "Looks", "Motion", and "Sound" blocks on one or more sprites on the same page.
<i>Flow control</i> Blocks are arranged in sequences	No sequences of blocks.	Sequence of blocks.	Repeats of sequences (use of the "Repeat" block, the "Repeat forever" and the "End" block). Includes "Trigger" block: start on tab.
<i>User interactivity</i> The user clicks on a sprite and triggers an action	There is no interactive capability.	Start on Green flag.	
<i>Data representation</i> Modifiers of sprite properties	No sprite properties have been modified.	Use of the "Shrink" and "Grow" blocks within a script with the default value (2).	Use of the "Shrink" and "Grow" blocks within a sequence of blocks using customized values (not the default value).

understand the narrative that the groups came up with and their efforts for decomposing and transferring this narrative to their storyboards and then to ScratchJr. During the storyboard interviews we also asked about children's impressions of the digital storytelling activity as well as their prior experiences with ScratchJr or any other programming tool. The average interview time was $\bar{x}=2:32$ ($SD=0:40$) for Cohort 1 and $\bar{x}=2:30$ ($SD=0:51$) for Cohort 2. Children's storyboard interviews were then selectively analyzed based on the results of the scoring of the CT level of the projects, to shed light on the existence of any quantitative differences between the two cohorts. As part of this study, we include two vignettes (one per cohort) to provide a more detailed picture of a representative group in each cohort.

Results

The 24 group projects produced a rich dataset, comprising of 94 pages, 388 ScratchJr sprites and 216 programming scripts. We present the findings as they relate to each of the research questions and conclude with a report of the level of CT exhibited in the children's projects. We begin with an overview of the data, which includes descriptive findings. Not all differences and comparisons were statistically significant; we indicate those cases in each of the sections. Qualitative data are presented as vignettes, to provide more nuanced answers to the research questions; they were selected to illustrate what the quantitative trends might mean. The interpretation of the findings can be found in the Discussion section.

Students' Coding Practices

With our first research question we sought to understand the main differences between the two cohorts' coding practices. Differences were detected in the average block usage by type of block for each cohort. The most used blocks by Cohort 1 (6–9-year-old children) were the "Motion" blocks, and the least used blocks were the "Control" blocks. On the other hand, the most used blocks by Cohort 2 (10–12-year-old children) were the "Looks" blocks and the least used blocks were the "Sound" blocks (see Figure 1).

As shown in Figure 1, children in each cohort mainly used the "Trigger", "Motion" and "Looks" blocks. Table 4 shows the average block usage by type of block per cohort.

As evident in Table 4 there were differences in the average block usage by type of block for each cohort. Subsequent tests employing the Mann-Whitney U test and Bonferroni correction indicated that children in Cohort 2 were more likely to employ "Trigger" blocks (Mann-Whitney, $U_{(22)}= 12.5$, $z=-3.31$, $p=.001$) when compared to children of Cohort 1. This was the only statistically significant difference identified.

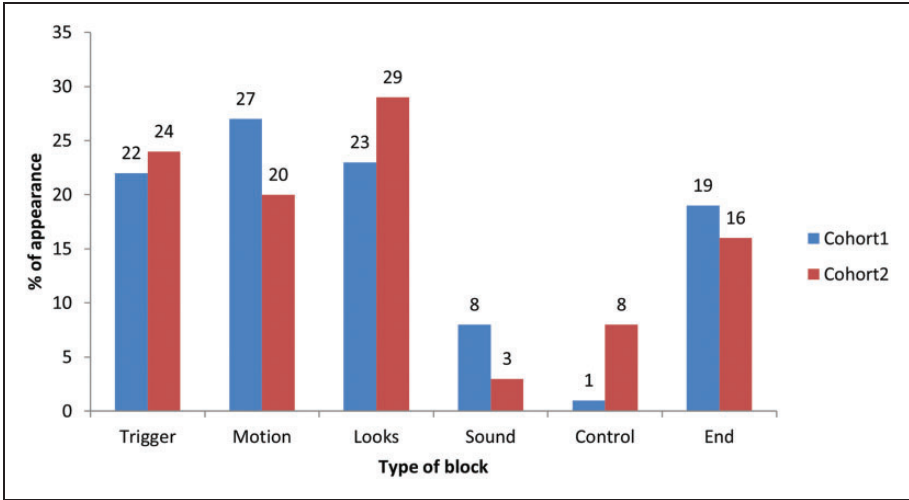


Figure 1. Percentage of Each Programming Block Employed in Children's Projects per Cohort.

Table 4. Average Block Usage by Type of Block per Cohort.

	Cohort 1 (n = 15 projects)		Cohort 2 (n = 9 projects)	
	Mean	SD	Mean	SD
Trigger blocks***	5.13	1.51	10.78	5.29
Motion blocks	6.33	7.55	9.00	5.92
Looks blocks	5.40	4.15	13.00	11.78
Sound blocks	1.80	1.97	1.33	2.06
Control flow blocks	.33	.62	3.56	5.79
End blocks	4.33	1.76	7.33	6.16

Note. *significant at the $p < 0.05$ level, **significant at the $p < 0.01$ level, ***significant at the $p < 0.001$ level.

Quality and Complexity of Children's ScratchJr Coding

To answer our second research question (RQ2) we identified the quality of children's collaborative projects by examining their complexity, based on the pages, sprites and scripts that were created. The ScratchJr indicators used were examined, such as for example, differences in unique backgrounds, inclusion of programmed sprites, or the number of functional scripts. Table 5 shows the descriptive statistics for each of the indicators per cohort.

Table 5. Pages, Sprites and Scripts per Cohort.

	Cohort 1 (n = 15 projects)		Cohort 2 (n = 9 projects)	
	Mean	SD	Mean	SD
Pages	3.93	.26	3.89	.33
Backgrounds	3.80	.56	3.44	1.33
Unique backgrounds*	2.87	1.13	1.67	1.00
Sprites*	12.60	5.55	22.11	9.28
Programmed sprites*	7.40	4.29	10.33	4.69
Non-programmed sprites	5.20	2.98	11.78	9.15
Default sprites	8.60	4.97	8.78	5.31
Edited sprites**	4	4.32	13.33	7.28
Scripts*	7.60	4.45	11.33	5.39
Functional scripts**	5.07	1.62	10.22	5.33
Total blocks*	23.33	9.80	45	31.14
Blocks with numbers**	2.67	2.85	8.78	7.99

Note. *significant at the $p < 0.05$ level, **significant at the $p < 0.01$ level, ***significant at the $p < 0.001$ level.

As shown in Table 5, there were differences in terms of the complexity and quality of children's projects between the two cohorts. Subsequent tests employing the Mann-Whitney U test and Bonferroni correction indicated that children in Cohort 1 were more likely to employ more unique backgrounds (Mann-Whitney, $U_{(22)} = 30$, $z = -2.30$, $p = .021$), when compared to projects completed by children of Cohort 2. However, children in Cohort 2 were more likely to have more sprites (Mann-Whitney, $U_{(22)} = 26$, $z = -2.48$, $p = .013$), programmed sprites (Mann-Whitney, $U_{(22)} = 31.50$, $z = -2.17$, $p = .030$), edited sprites (Mann-Whitney, $U_{(22)} = 21.5$, $z = -2.77$, $p = .006$), scripts (Mann-Whitney, $U_{(22)} = 30$, $z = -2.28$, $p = .023$), functional scripts (Mann-Whitney, $U_{(22)} = 17.5$, $z = -3.02$, $p = .003$), number of total blocks (Mann-Whitney, $U_{(22)} = 31.50$, $z = -2.15$, $p = .032$), and blocks with numbers (Mann-Whitney, $U_{(22)} = 22.50$, $z = -2.70$, $p = .007$) when compared to children of Cohort 1.

Children's CT Concepts

With RQ3 we sought to understand the main differences in CT between the cohorts. Figures 2A to 2C show the percentage of projects in each cohort for each of the three levels of the six CT concepts, as assessed in this study.

As shown in Figures 2A to 2C, several Cohort 2 projects reached higher levels of CT than Cohort 1 on four of the six CT concepts. Projects in both cohorts only reached Level 1 for CT in the categories of *flow control* and *user interactivity*; furthermore, the fact that the group projects in both cohorts were scored at Level 0 for the categories of *parallelism*, *synchronization*, and *data*

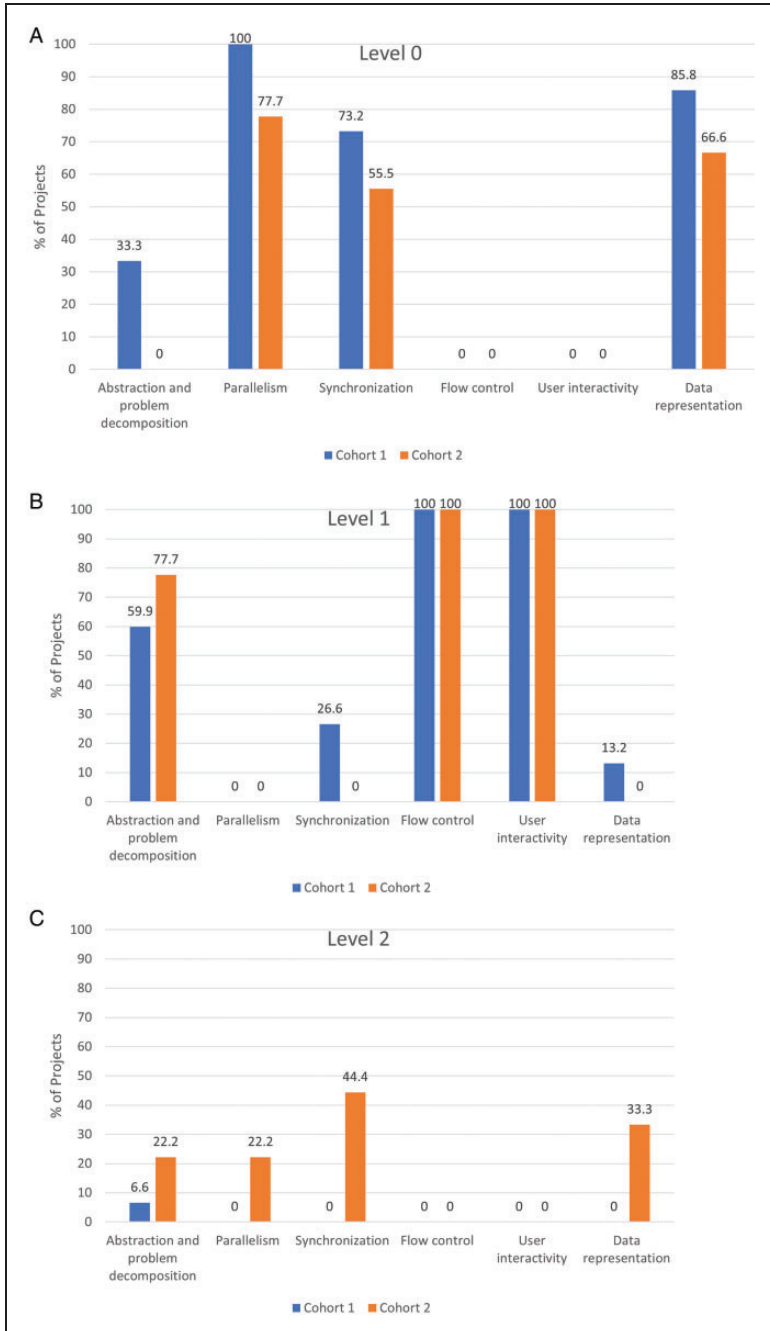


Figure 2. A–C: Percentage of Projects per Cohort Exhibiting Each Level of CT (L0 = Lowest Level, L2 = Highest Level).

representation indicates that these CT concepts were challenging for these primary school children, even though they were slightly less challenging for Cohort 2 than for Cohort 1.

In the category of *Abstraction and problem decomposition*, Cohort 2 fared better than Cohort 1, by including more than one sprite and more than one script in their pages and involving more sprites in their stories which were then coded to act a series of events. Figure 3 provides a screenshot of one of the pages in the project created by Group A (Cohort 1). This project was evaluated as belonging to Level 2 of *Abstraction and problem decomposition*, which was the highest possible level. The children in Group A created a story that proposed addressing waste management through the restoration of a polluted city park. The examination of the group’s coding indicates that the children programmed more than one sprite. Figure 3 presents the script of the main sprite in this page; the script is comprised of a sequence of three blocks, indicating a script of low complexity, which was used to provide the main sprite (the cat) limited motion (only two steps forward).

Figure 4 shows a screenshot of one of the pages created by Group B, which is more typical of the remaining projects in Cohort 1, which were evaluated at Level 1 or Level 0 for the category “*Abstraction and problem decomposition*”. This group’s project was scored as belonging to Level 1, as the project did not exhibit sophisticated problem decomposition through using multiple scripts and sprites in each of their project’s page.



Figure 3. Group A ScratchJr Project (Cohort 1).

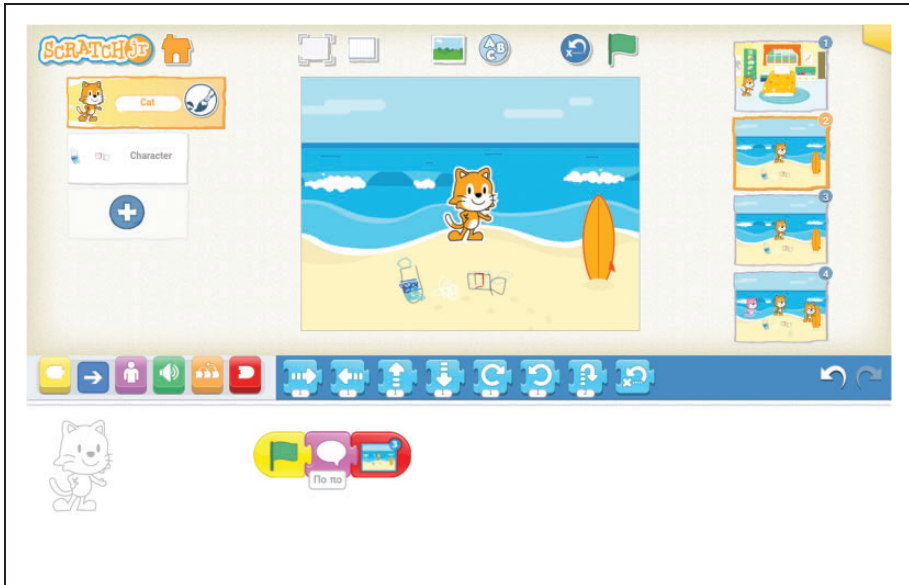


Figure 4. Group B ScratchJr Project (Cohort 1).

In the category of *Parallelism*, none of the projects in Cohort 1 included a coded sprite with two scripts running in parallel, as children only coded their sprite-related events to run sequentially. Coding a sprite to respond to two or more actions at the same time shows higher complexity and sophistication, may require higher levels of CT, and thus, more advanced capabilities of coding.

Similarly, there is an observed difference between the two cohorts in the *Synchronization* category. Synchronizing the actions between two or more sprites demands a logical sequence of those actions in order for the story to make sense, but most importantly, it demands excellent synchronization of those actions. None of Cohort's 1 projects were evaluated as Level 2 for this category. The comparison of the *Flow control* and *User interactivity* categories showed no differences between the two cohorts. Both cohorts used a sequence of blocks to code their sprites for an action and included the green flag to trigger those actions.

The projects in the two cohorts also differed in the *Data representation* category. Cohort 1 projects barely employed the "Shrink" and "Grow" blocks within a script, even though children in this cohort often used those blocks to change the representation of a sprite independently of a sequence of actions. For example, they drew rubbish and used the "Shrink" block to resize the rubbish in order to fit in a bin but did not include any other coded action beyond this and



Figure 5. A Page From Group C Project (Cohort 2).

did not integrate these blocks in their script. On most occasions, the block was deleted after the action. On the other hand, Cohort 2 projects employed those two blocks to give the feeling of three-dimensional space as the sprites moved around the stage. Figure 5 shows one page from Group C's project (Cohort 2).

Children in Group C programmed more than one sprite in their story. The script presented in Figure 5 was comprised of a sequence of fourteen blocks, indicating a script of high complexity that was employed for assigning the main ScratchJr sprite (the cat) with motion, words (using the "Say" blocks), show and hide properties, as well as time control of these actions. The analysis of the CT concepts indicates that this group performed at Level 1 for the "*Abstraction and problem decomposition*" category, as the children employed more than one script and more than one sprite in their page, Level 2 for the "*Synchronization*" category, as the children employed the "Control" blocks for coordinating the events among their sprites, Level 1 for the "*Flow control*" category, as the blocks were placed in a sequence and "Motion" blocks were also numbered for achieving a better flow, and Level 1 for the "*Data representation*" category, as the children employed the "Looks" blocks for modifying the properties of the sprite. The project of Group C was at the lowest level (L0) for the "*Data representation*" category, which concerns dynamic representations of data which are programmed and indicates sophisticated skills in coding. Figure 6 shows the percentage of the projects' total scores per CT concept in relation to the maximum score that the children could reach, for each of the two cohorts.

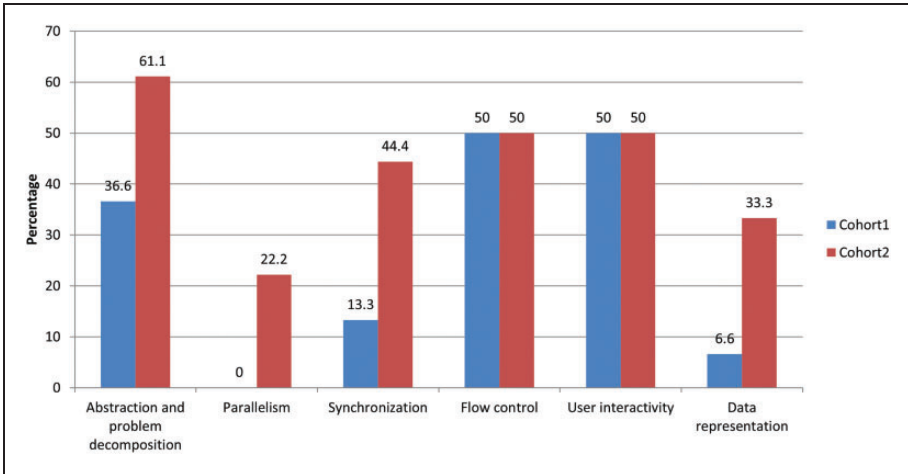


Figure 6. Scores per Computational Concept.

The percentage of the projects' total scores for *Flow control* and *User Interactivity* in relation to the maximum score the children could achieve, was the same for both cohorts. However, the percentage of the projects' total scores for the remaining CT concepts (i.e., *Abstraction and problem decomposition*, *Parallelism*, *Synchronization*, *Data representation*) in relation to the maximum score the children could obtain, was larger for Cohort 2.

Finally, the average overall score and score per computational concept category, as shown in Table 6, indicate a trend between the two cohorts, with Cohort 2 outperforming Cohort 1 in *Parallelism*, *Synchronization* and *Data representation*. However, statistically significant differences were only identified for the computational concept of *Abstraction and problem decomposition*. In particular, a Mann-Whitney U test indicated that Cohort 2 outperformed Cohort 1 and that this difference was statistically significant (Mann-Whitney, $U_{(22)}=25.5$, $z=-2.62$, $p=.009$), with Cohort 2 outperforming Cohort 1 (Mann-Whitney, $U_{(22)}=39.5$, $z=-2.00$, $p=.045$) in regard to the category of *Abstraction and problem decomposition*.

Students' Use of CT to Create a Digital Story

Our fourth research question concerned whether the groups were able to apply CT to create a digital story about an environmental issue through coding in ScratchJr. Despite the documented shortcomings, all groups were successful in conveying a basic message about environmental waste management through the stories they coded. We next present two vignettes, to provide a more detailed picture of one representative group from each cohort, using the cases of Joanna

Table 6. Average Overall Score and Score per Computational Concept Category.

	Cohort 1 (n = 15 projects)		Cohort 2 (n = 9 projects)	
	Mean	SD	Mean	SD
Total score**	3.13	.743	5.22	2.39
Abstraction and problem decomposition*	.73	.59	1.22	.44
Parallelism	.00	.00	.44	.88
Synchronization	.27	.46	.89	1.05
Flow control	1.00	.00	1.00	.00
User interactivity	1.00	.00	1.00	.00
Data representation	.13	.35	.67	1.00

Note. *significant at the $p < 0.05$ level, **significant at the $p < 0.01$ level, ***significant at the $p < 0.001$ level.

and Iris (Cohort 1) and Anna and Marie (Cohort 2) [all names are pseudonyms]. We focus on the CT concept of “*Abstract and problem decomposition*” since that was the only one for which we found statistical differences between the two cohorts. In particular, the vignette of Joanna and Iris is more typical of the projects in Cohort 1, which were mostly evaluated as Level 1 or Level 0 for the *Abstraction and problem decomposition* concept, reflecting less advanced CT practices. Likewise, the vignette of Anna and Marie is more typical of the projects in Cohort 2, which were mostly evaluated as Level 1 or Level 2 for the category *Abstraction and problem decomposition*, thus, reflecting more advanced CT practices.

Vignette 1: The Case of Joanna and Iris [Cohort 1 – Total Score 2/12]. The story Joanna and Iris coded was titled “The dirty field land” and had two friends (a young boy and a young girl) as main characters. According to their narrative, as told by Joanna (a six-year-old girl) in her interview:

Once upon a time there was a young girl who was heading to school. On her way she found a young boy, her best friend, cleaning up a dirty field. Along with other school mates, he helped his friend to clean up the field. Later on, they saw another young boy throwing garbage in the same field. They cleaned up once again and explained to the young boy that it is wrong to throw garbage in the field. The young boy understood his mistake and he never dropped garbage in the field ever since.

The environmental message that Joanna and Iris tried to communicate through their story was that “*we should not throw our garbage in the field*” [Iris, 6 years

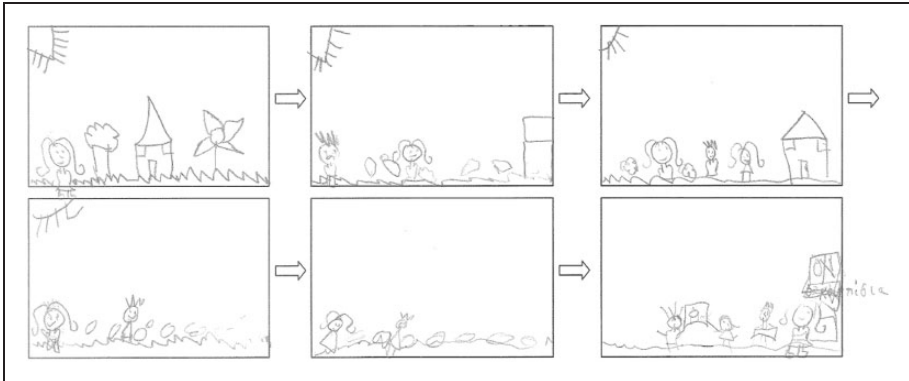


Figure 7. Joanna and Iris' Storyboard.

old], aimed at increasing awareness about keeping the earth clean by avoiding garbage disposal in non-appropriate places.

After finalizing their story idea, the two girls proceeded with the first step of decomposing the environmental story using the storyboard they had created (Figure 7). According to their storyboard, they divided their story in scenes; in each scene they added the background (e.g., the field, the school yard), main characters (the two friends, the third kid, their schoolmates) and the different objects (e.g., the garbage, the house, the school).

They then transferred and coded their story in ScratchJr. Overall, Joanna and Iris were successful in transferring the story scenes from the storyboard in ScratchJr. However, as evident in Figure 8, the girls did not exhibit sophisticated problem decomposition by using multiple scripts and sprites in each of their project's page.

The scripts employed were usually quite short (up to 4 blocks) and as presented in Figure 9, in some cases the scripts were composed only by one block, and were not functional, as they were lacking the "Trigger" and the "End" blocks.

Vignette 2: The Case of Anna and Marie [Cohort 2 –Total Score 10/12]. The story of Anna and Marie (both 11 years old) was titled "Garbage in our yard" and included several main characters: the mom, the dad, the daughter, their dog and their horse. The following narrative was used, as described by the children in their interviews:

Once upon a time this family discovered tons of garbage in their yard. And they wondered: who did this to us? But after a while they discovered that they were the ones who were responsible for this mess. But how could they solve this problem?

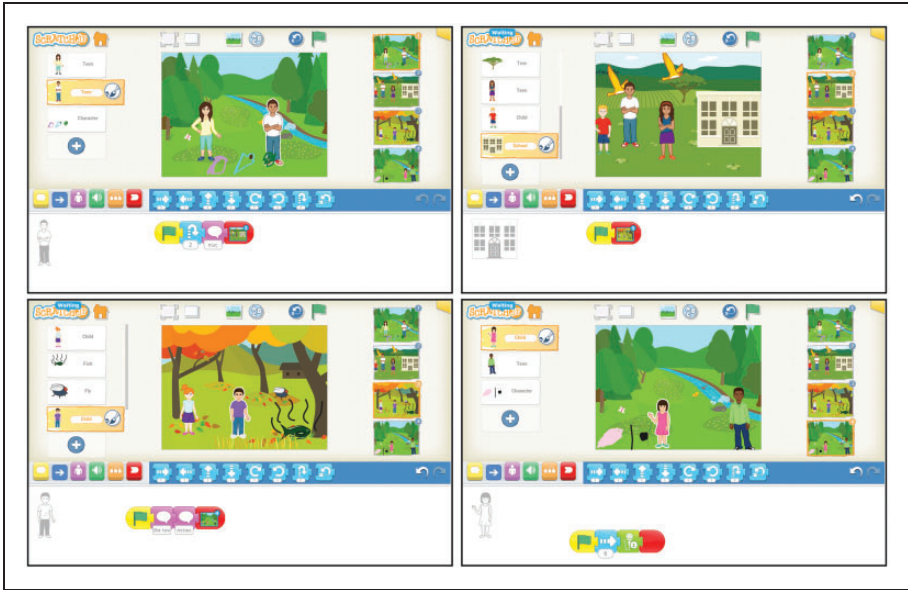


Figure 8. Problem Decomposition in ScratchJr by a Cohort I Group (Joanna/Iris).

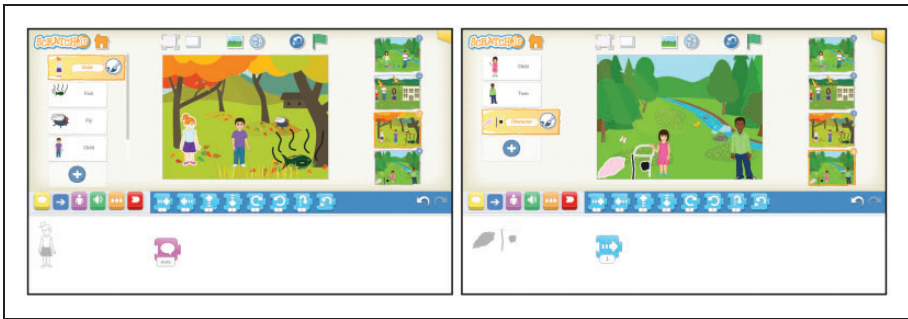


Figure 9. Examples of Non-Functional Scripts.

They cleaned up their yard but after two days, the situation was the same. At the end they decided to start recycling [Marie, 11 years old].

The environmental message that they tried to communicate through their story was that “People consume a lot, but they do not compost or recycle the garbage they produce. They should recycle to reduce waste” [Anna, 11 years old], and aimed at increasing awareness about waste management actions, such as recycling.

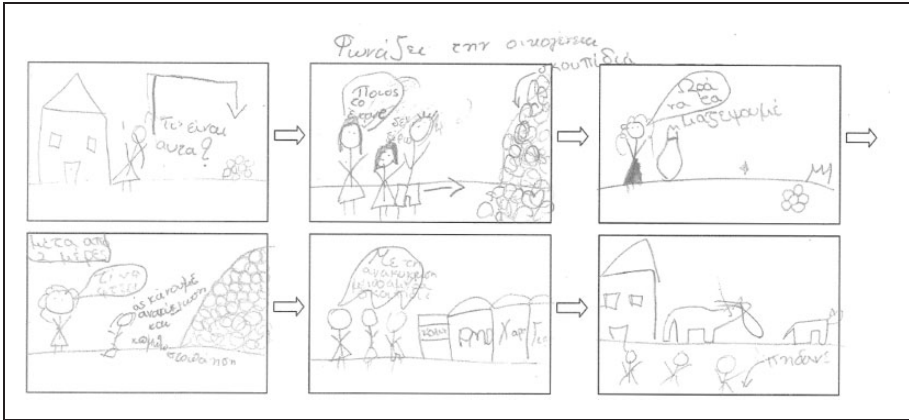


Figure 10. Anna and Marie's Storyboard.

After conceptualizing their story idea and finalizing their storyboard, the two girls proceeded with the first step of decomposing the environmental story (Figure 10). As in the previous vignette, the two girls divided their story in scenes; in each scene they added the backgrounds (e.g., the farmhouse, the yard), the main characters (the mother, the father, the daughter) and different objects (e.g., the farm animals, the garbage, the recycling bins).

They next transferred and coded their story in ScratchJr, attempting to add interactivity to the narrative. Overall, Anna and Marie were successful in transferring the story scenes from the storyboard to ScratchJr (Figure 11).

In addition, the two girls in this vignette exhibited a sophisticated problem decomposition as they used multiple scripts and spites in each of their project's pages. For instance, as shown in Figure 12, all characters (mother, father, daughter and dog) were scripted. Each script was fully functional, contained at least 4 blocks, and was comprised of different combinations of blocks ("Trigger", "Motion", "Looks", "Control flow" and "End" blocks), as well as by blocks with numbers.

Discussion

This study investigated primary school children's collaborative coding to create a digital story in response to an environmental problem. We purposefully chose two age cohorts, in order to characterize students' practices at two different points in their development and to begin understanding the developmental trajectory of what it means to engage in CT and how it may be applied to and manifested in the students' artifacts. The contrast between the age cohorts can provide a first glimpse into similarities and differences between these ages and can guide work on how to support the development of younger children's CT.



Figure 11. Problem Decomposition by a Cohort 2 Group (Anna/Marie).



Figure 12. Examples of Scripts and Blocks Employed by Anna and Marie.

Findings indicated that children were successful in using the ScratchJr software to convey their own environmental science messages about effective waste management actions. At the same time, the analysis of the projects from both cohorts revealed similarities in children's coding practices and underlying CT concepts. The findings of this study also point to possible age differences in what children may be initially able to do with coding environments such as ScratchJr. These findings suggest that the "low threshold" idea was successful for these children (Flannery et al., 2013), but that to achieve "high ceiling" one needs to experience scaffolded activities that focus on coding and CT challenges for the children in developmentally appropriate ways.

We next discuss these findings as they relate to the children's coding practices, and their respective CT.

Children's Coding Practices

Some of the results reported in our study about children's coding practices are similar in nature to prior studies focusing on ScratchJr (e.g., Papadakis et al., 2016; Portelance et al., 2016). For instance, in alignment with Papadakis et al. (2016), we also found that the most used blocks by Cohort 1 (6–9-year-old children) were the "Motion" blocks, and the least used blocks were the "Control flow" blocks. As in prior studies, these results could be attributed to the more intuitive nature of the "Motion" blocks and the more complex understanding children would need to possess to be able to use the "Control flow" blocks (Portelance et al., 2016; Strawhacker & Bers, 2019). Such results strengthen findings that suggest a trajectory of learning that is also influenced by maturation and development (Flannery & Bers, 2013).

On the other hand, we have found that the most used blocks by Cohort 2 (10–12 years-old) were the "Looks" blocks which, according to our knowledge, is a finding not reported in previous studies. This finding was demonstrated by this cohort's emphasis on adding dialog boxes in their storytelling, using the "Say" blocks. At the same time, other findings of this study, such as the greater use of "Trigger" blocks by older children (Cohort 2), seem to differ from findings reported by Portelance et al. (2016), who reported that older children used significantly fewer "Trigger" blocks than younger ones. Finally, in our study younger children used the "Sound" blocks to a greater degree. A potential explanation could be that younger children preferred to add narrative in their digital stories using the recording feature (due to their limited writing skills), in contrast to the older children who added narrative using the "Looks" dialogue blocks. The statistically significant difference observed in the use of "Trigger" blocks can be linked to the number of scripts developed by the children in each cohort, as children in Cohort 2 created more scripts than the children in Cohort 1. Nonetheless, these results need to be interpreted with caution due to the small dataset of the present study and due to the nature of the learning task with

ScratchJr. It could be that other problem-solving contexts may lead to different behaviors.

We also identified several differences in the complexity and quality of children's projects between the two cohorts. In particular, children in Cohort 2 were more likely to have more sprites, programmed sprites, edited sprites, scripts, functional scripts, number of total blocks and blocks with numbers, when compared to children of Cohort 1. These findings are also aligned with prior studies which provided empirical substantiation supporting that older children encountered fewer difficulties, which resulted in more complex and functional scripts. For instance, Papadakis et al. (2016) reported that younger children encountered difficulties when the scene included the coding of more than one sprite, while according to Portelance et al. (2016) older children were more efficient in coding and coordinating multiple sprites. These findings also point to developmental differences and the possibility of a developmental trajectory in younger children's ability to code. These results are not surprising, given what we know about how learning develops. However, they point to the need to investigate what develops and how, and the extent to which scaffolding could help children move within their zone of proximal development (Vygotsky, 1980) in regard to CT.

Children's CT

Prior programming studies have primarily focused on the investigation of the children's coding practices (e.g., Papadakis et al., 2016; Portelance et al., 2016). Extending these efforts, the present study is also focused on the investigation of the CT concepts underlying children's coding practices. According to our findings, the overall projects' scores indicated trends between the two cohorts, with Cohort 2 outperforming Cohort 1 in *Abstraction & problem decomposition*, *Parallelism*, *Synchronization* and *Data representation*. These findings could be attributed to the children's cognitive development. As supported by Strawhacker and Bers (2019) prior spatial and causal reasoning, and children's cognitive development (Goswami & Bryant, 2012), could influence children's computational practices in a given task and could provide a potential explanation of the differences observed in children's CT. Nonetheless, this developmental perspective seems to be confounded with other challenges that may not be solely age- and development-dependent: according to our findings significant differences were only identified for the computational concept of *Abstraction & problem decomposition*. Specifically, it appears that younger children appear to need more support to engage with the CT concept of *abstraction & problem decomposition*.

Perhaps what is most interesting to consider in thinking about how to support students' development of CT is not the differences between the cohorts but the similarities. Most notable in this is the similar low performance of student

groups in terms of the remaining five computational concepts. Very few projects were assessed at Level 2, whereas projects were assessed poorly at Level 0 for three of the six computational concepts (Parallelism, Synchronization, and Data representation). The differences between the assessment of the computational concepts suggest that various challenges might be at play; it seems promising to explore each of the computational concepts in more depth in future studies, so that we can understand the specific challenges as well as how children could be supported in overcoming them.

Limitations and Future Work

Even though the findings of this study contribute to a better understanding of primary school children's coding practices and CT, having adopted a developmental perspective, it is important to note the limitations of this work. First, the sample participating in the study was relatively small. While this sample is deemed appropriate, given the exploratory nature of this study, future studies should aim for a larger sample which would allow systematic comparison between more age groups. Another limitation of this work might be the heavy use of the children's projects for the data analysis; even though the analysis of children's projects is an example of embedded assessment, and as such it is valued for the rich data it can provide and the minimal intrusiveness into the children's activity, collecting data on children's understanding of the specific CT concepts can provide more insights in reasoning processes and practices.

An additional limitation might be that the data we reported on were the product of children's collaboration and as such they were analyzed at the group level and not at an individual level. In future research, individual assessment of CT, in addition to project artifact analysis, can provide richer access to individual children's thinking and analytical reasoning.

Our study was purposefully conducted as a cross-sectional study. However, the lack of baseline assessments and pre-posttests did not allow for an examination of within-group development in terms of coding practices, CT skills, and students' conceptual understanding of the environmental topic. Other personal characteristics, such as children's literacy skills, may also affect their coding practices and preferences (e.g., younger children preferred to audio-record their narratives, while older children opted in communicating across their coded stories using written texts). Such factors can be investigated in future studies.

Finally, our findings are most relevant to children's coding practices and CT as these unfolded in the context of a specific task (i.e., development of a digital story). Future studies could investigate and even compare whether and how children's coding practices and CT may differ in the context of various tasks in ScratchJr.

Conclusions and Implications

This study set out to investigate differences and similarities in the use of computational elements in ScratchJr between two age cohorts of primary school children in grades 1-6. Findings contribute to empirical data on how two cohorts of primary school children of different ages, exhibited coding practices and CT. The findings of this study confirmed past research but also provided additional insights regarding the quality of the children's work and a possible learning trajectory, via the comparison of lower and upper primary school children's coding practices and CT. Based on these findings, it appears that children's collaborative performance with coding is aligned with children's age, as children's projects increased in sophistication from Cohort 1 (younger children) to Cohort 2 (older children). However, these findings also have educational and theoretical implications.

At a first glimpse, the findings of the present study may not seem surprising as they could be attributed to children's developmental differences. However, as argued by Strawhacker and Bers (2019), although the "trajectory of improvement over successive grades is a familiar pattern" (p. 563), the domain of programming is an emerging one, providing a new study area for developmental researchers. We cannot take for granted that developmental trajectories in traditional academic areas will perfectly fit emerging learning areas such as coding and CT. Instead, we need to delve deeper in the investigation of children's learning trajectories in the field of CT, as a unique and differentiated area that deserves to be studied on its own.

Based on the findings of this research work, it seems that it is worth adopting a developmental approach and it is worth describing children's coding practices and CT as a learning progression. Therefore, our study implies that, from a developmental point of view, the introduction of coding and CT requires educational activities which consider children's cognitive affordances and limitations (see also Bers, 2017, 2019). At the same time, based on the Vygotskian theory of the zone of proximal development, our findings provide useful insights especially regarding the youngest children's learning challenges in coding and CT, while using ScratchJr. These findings imply that pairing children from the two age cohorts in coding activities could serve as a scaffolding mechanism to bridge the distance between what a young learner can do on his/her own and what s/he can achieve with the guidance and encouragement from a skilled partner when coding. While previous studies have investigated pairing parents and children in the context of coding activities (Govind et al., 2020) future studies could also focus on pairing children of different ages and developmental stages, and thus, of different coding expertise.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Ethical Approval


All procedures performed in the study involving human participants were in accordance with the ethical standards of the institutional research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.


Informed Consent

Informed consent was obtained from all participants included in the study, and their legal guardians.

ORCID iDs

Eleni A. Kyza  <https://orcid.org/0000-0003-0992-4034>

Yiannis Georgiou  <https://orcid.org/0000-0002-2850-8848>

Andria Agesilaou  <https://orcid.org/0000-0002-9943-7223>

Markos Souropetsis  <https://orcid.org/0000-0003-3647-2272>

Supplemental material

Supplemental material for this article is available online.

References

- Ala-Mutka, K. (2004). Problems in learning and teaching programming—A literature study for developing visualizations in the Codewitz-Minerva project. *Codewitz Needs Analysis*, 20.
- Barcelos, T. S., Muñoz-Soto, R., Villarroel, R., Merino, E., & Silveira, I. F. (2018). Mathematics learning through computational thinking activities: A systematic literature review. *Journal of Universal Computer Science*, 24(7), 815–845.
- Barrouillet, P., & Lecas, J. F. (1999). Mental models in conditional reasoning and working memory. *Thinking & Reasoning*, 5(4), 289–302.
- Baser, M. (2013). Attitude, gender and achievement in computer programming. *Middle-East Journal of Scientific Research*, 14(2), 248–255.
- Baytak, A., & Land, S. M. (2011). An investigation of the artifacts and process of constructing computers games about environmental science in a fifth-grade classroom. *Educational Technology Research and Development*, 59(6), 765–782.
- Bers, M. U. (2012). *Designing digital experiences for positive youth development: From playpen to playground*. Oxford University Press.
- Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.
- 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.
- 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.

- Bers, M. U., & Resnick, M. (2015). *The official ScratchJr book*. No Starch Press, Inc.
- Brennan, K., Chung, M., & Hawson, J. (2011). *Creative computing: A design-based introduction to computational thinking*. <https://scratched.gse.harvard.edu/sites/default/files/curriculumguide-v20110923.pdf>
- Brennan, K., & Resnick, M. (2012, April 13–17). *New frameworks for studying and assessing the development of computational thinking* [Paper presentation]. American Educational Research Association annual meeting (AERA 2012), Vancouver, Canada.
- Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87(4), 834–860.
- Cadorna, E. A., Cadorna, E. F., & Taban, J. G. (2021). A Cross-Sectional study of students' learning progression in algebra. *Universal Journal of Educational Research*, 9(3), 449–460.
- Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95, 202–215.
- Chou, P. N. (2020). Using ScratchJr to foster young children's computational thinking competence: A case study in a third-grade computer class. *Journal of Educational Computing Research*, 58(3), 570–595.
- Clarke-Midura, J., Lee, V. R., Shumway, J. F., & Hamilton, M. M. (2019). The building blocks of coding: A comparison of early childhood coding toys. *Information and Learning Sciences*, 120(7/8), 505–518.
- Clements, D. H. (1999). The future of educational computing research: The case of computer programming. *Information Technology in Childhood Education Annual*, 1, 147–179.
- Creswell, J. W., & Clark, V. L. P. (2007). *Designing and conducting mixed methods research*. SAGE Publications Inc.
- Cummings, C. L. (2018). *Cross-sectional design. The SAGE encyclopedia of communication research methods*. SAGE Publications Inc.
- de Araujo, A. L. S. O., Andrade, W. L., & Guerrero, D. D. S. (2016, October 12–15). *A systematic mapping study on assessing computational thinking abilities* [Paper presentation]. IEEE frontiers in education conference (FIE) 2016 (pp. 1–9). IEEE.
- Duncan, R. G., Rogat, A., & Yarden, A. (2009). A learning progression for deepening students' understanding of modern genetics across the 5th-12th grades. *Journal of Research in Science Teaching*, 46(6), 655–674.
- Durak, H. (2016). *Design and development of an instructional program for teaching programming process to gifted students* [Unpublished doctoral dissertation]. Gazi University.
- Durak, H. Y. (2020). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning*, 25(1), 179–195.
- Duschl, R. A. (2019). Learning progressions: Framing and designing coherent sequences for STEM education. *Disciplinary and Interdisciplinary Science Education Research*, 1(1), 10.
- Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63, 87–97.

- Fields, D., Vasudevan, V., & Kafai, Y. B. (2015). The programmers' collective: Fostering participatory culture by making music videos in a high school scratch coding workshop. *Interactive Learning Environments*, 23(5), 613–633.
- Flannery, L. P., & Bers, M. U. (2013). Let's dance the "robot hokey-pokey!": Children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education*, 46(1), 81–101.
- Flannery, L. P., Kazakoff, E. R., Bontá, P., Silverman, B., Bers, M. U., & Resnick, M. (2013). *Designing ScratchJr: Support for early childhood learning through computer programming* [Paper presentation]. ACM International Conference Proceeding Series (pp. 1–10). ACM.
- Flavell, J. H., Miller, P. H., & Miller, S. A. (1993). *Cognitive development* (3rd ed.). Prentice Hall.
- Fokides, E. (2018). Teaching basic programming concepts to young primary school students using tablets: Results. *International Journal of Mobile and Blended Learning*, 10(1), 34–47.
- Goswami, U., & Bryant, P. (2012). Children's cognitive development and learning. In R. Alexander, C. Doddington, J. Gray, L. Hargreaves, & R. Kershner (Eds.), *The Cambridge primary review research surveys* (pp. 161–189). Routledge.
- Govind, M., Relkin, E., & Bers, M. U. (2020). Engaging children and parents to code together using the ScratchJr app. *Visitor Studies*, 23(1), 46–65.
- Grover, S., & Pea, R. (2013). Computational thinking in K–12. A review of the state of the field. *Educational Researcher*, 42(1), 38–43.
- Heintz, F., Mannila, L., & Färnqvist, T. (2016, October 12–15). *A review of models for introducing computational thinking, computer science and computing in K-12 education* [Paper presentation]. Frontiers in Education Conference (FIE) 2016 (pp. 1–9). IEEE.
- Hermans, F., & Aivaloglou, E. (2017, November 8–10). *To scratch or not to scratch? A controlled experiment comparing plugged first and unplugged first programming lessons* [Paper presentation]. 12th workshop on primary and secondary computing education, Nijmegen, Netherlands.
- Hill, C., Dwyer, H. A., Martinez, T., Harlow, D., & Franklin, D. (2015, March 4–7). *Floors and flexibility. Designing a programming environment for 4th-6th grade classrooms* [Paper presentation]. 46th ACM Technical Symposium on Computer Science Education (pp. 546–551), Kansas City, USA. ACM.
- 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.
- Hu, Y., Chen, C. H., & Su, C. Y. (2021). Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis. *Journal of Educational Computing Research*, 58(8), 1467–1493.
- Jakoš, F., & Verber, D. (2017). Learning basic programming skills with educational games: A case of primary schools in Slovenia. *Journal of Educational Computing Research*, 55(5), 673–698.
- Jung, S., & Won, E. S. (2018). Systematic review of research trends in robotics education for young children. *Sustainability*, 10(4), 905.
- Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press.

- Kong, S. C., & Wang, Y. Q. (2019, November 7–8). *Assessing programming concepts in the visual block-based programming course for primary school students* [Paper presentation]. 18th European Conference on e-Learning, ECEL 2019, Copenhagen, Denmark.
- Kong, S. C., & Wang, Y. Q. (2021). Item response analysis of computational thinking practices: Test characteristics and students' learning abilities in visual programming contexts. *Computers in Human Behavior*, *122*, 106836.
- Kraleva, R., Kralev, V., & Kostadinova, D. (2019). A methodology for the analysis of block-based programming languages appropriate for children. *Journal of Computing Science and Engineering*, *13*(1), 1–10.
- Lee, K. T., Sullivan, A., & Bers, M. U. (2013). Collaboration by design: Using robotics to foster social interaction in kindergarten. *Computers in the Schools*, *30*(3), 271–281.
- Lindberg, R. S., Laine, T. H., & Haaranen, L. (2019). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games. *British Journal of Educational Technology*, *50*(4), 1979–1995.
- 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.
- Mioduser, D., Levy, S. T., & Talis, V. (2009). Episodes to scripts to rules: Concrete-abstracts in kindergarten children's explanations of a robot's behavior. *International Journal of Technology and Design Education*, *19*(1), 15–36.
- Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies*, *23*(4), 1483–1500.
- Moors, L., Luxton-Reilly, A., & Denny, P. (2018, April 19–22). *Transitioning from block-based to text-based programming languages* [Paper presentation]. International Conference on Learning and Teaching in Computing and Engineering (LaTICE), Auckland, New Zealand.
- Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *Revista de Educación a Distancia*, *46*, 1–23.
- Müller, U., Overton, W. F., & Reese, K. (2001). Development of conditional reasoning: A longitudinal study. *Journal of Cognition and Development*, *2*(1), 27–49.
- National Research Council. (2011). *Report of a workshop on the pedagogical aspects of computational thinking*. National Academies Press. <https://www.nap.edu/catalog/13170/report-of-a-workshop-on-the-pedagogical-aspects-of-computational-thinking>
- Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, *11*(1), 1–17.
- Papadakis, S. (2020). Apps to promote computational thinking concepts and coding skills in children of preschool and pre-primary school age. *Mobile Learning Applications in Early Childhood Education* (pp. 101–121). IGI Global.
- 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>.

- Pea, R. D. (1986). Language-independent conceptual “bugs” in novice programming. *Journal of Educational Computing Research*, 2(1), 25–36.
- 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.
- Portelance, D. J. (2015). Code and tell. *An exploration of peer interviews and computational thinking with ScratchJr in the early childhood classroom* [Doctoral dissertation, Tufts University]. ProQuest Dissertations and Theses Global.
- Portelance, D. J., & Bers, M. U. (2015, June 21–25). *Code and tell. Assessing young children’s learning of computational thinking using peer video interviews with ScratchJr* [Paper presentation]. The 14th International Conference on Interaction Design and Children, Medford, OR, USA.
- 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.
- 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.
- Resnick, M. (2013, May 8). *Learn to code, code to learn: How programming prepares kids for more than math*. EdSurge. <https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn>
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67.
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137–172.
- Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two-year case study using “scratch” in five schools. *Computers & Education*, 97, 129–141.
- ScratchJr.org. (2015a). *Interface guide*. <http://www.scratchjr.org/>
- ScratchJr.org. (2015b). *About ScratchJr*. <http://www.scratchjr.org/about.html>
- Shodiev, H. (2015). Computational thinking and simulation in teaching science and mathematics. In M. G., Cojocar, I. S., Kotsireas, R. N., Makarov, R. V., Melnik, & H., Shodiev (Eds.), *Interdisciplinary topics in applied mathematics, modeling and computational science*. Springer.
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158.
- 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(3), 541–575.
- Strawhacker, A., Lee, M., Caine, C., & Bers, M. (2015, June). ScratchJr Demo: A coding language for Kindergarten. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 414–417).
- Sullivan, F. R., & Heffernan, J. (2016). Robotic construction kits as computational manipulatives for learning in the STEM disciplines. *Journal of Research on Technology in Education*, 48(2), 105–128.

- Sun, L., Hu, L., & Zhou, D. (2021). 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*. Advance online publication. <https://doi.org/10.1111/jcal.12545>
- Tran, Y. (2019). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research*, *57*(1), 3–31.
- Vasilopoulos, I. V., & Van Schaik, P. (2019). Koios: Design, development, and evaluation of an educational visual tool for Greek novice programmers. *Journal of Educational Computing Research*, *57*(5), 1227–1259.
- Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Harvard University Press.
- Wilson, M., & Sloane, K. (2000). From principles to practice: An embedded assessment system. *Applied Measurement in Education*, *13*(2), 181–208.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.
- Yu, J., & Roque, R. (2019). A review of computational toys and kits for young children. *International Journal of Child-Computer Interaction*, *21*, 17–36.
- Žanko, Ž., Mladenović, M., & Boljat, I. (2019). Misconceptions about variables at the K-12 level. *Education and Information Technologies*, *24*(2), 1251–1268.

Author Biographies

Eleni A. Kyza is Associate Professor in Information Society at the Department of Communication and Internet Studies at the Cyprus University of Technology, where she coordinates the *Media, Cognition, and Learning Research Group*. Her research focuses on the investigation of technology-enhanced learning environments to support motivated, meaningful, and reflective practices, and the investigation of how new media influence human behavior. Her work has addressed, among others, issues of inquiry-based learning, teacher professional development, scaffolding student learning, collaborative learning, and media & information literacy on social media. With her colleagues, she has developed and empirically investigated learning technologies, such as the web-based learning and teaching platform *STOCHASMOS* for promoting evidence-based reasoning in science education, and *TraceReaders*, an augmented reality platform for scaffolding students' inquiry learning in informal and non-formal contexts.

Yiannis Georgiou is a research fellow with the *Media, Cognition, and Learning Research Group* at the Department of Communication and Internet Studies of the Cyprus University of Technology. He holds a bachelor's degree in Elementary School Teaching from the University of Cyprus, a master's degree in Science & Environmental Education from the University of Cyprus, and a PhD in Communication and Internet Studies from the Cyprus University of Technology. His research interests are focused on the investigation of emerging

technologies for learning as well as on teachers' professional development in relation to novel pedagogies and technologies.

Andria Agesilaou is a PhD candidate and a research associate with the Department of Communication and Internet Studies at the Cyprus University of Technology. She has a Master's degree in "New Technologies for Communication and Learning" from the same department, and a BA in Primary Education from the National and Kapodistrian University of Athens, Greece. Her current research interests focus on the design and development of technology-enhanced learning experiences for students. Her work investigates how to empower students to develop critical understanding of their personal data.

Markos Souropetsis holds an MSc in Cultural Informatics and Communication and a BSc in Cultural Technology and Communication, both from the University of the Aegean, Greece. He is a PhD candidate with the Department of Communication and Internet Studies at the Cyprus University of Technology. His research focuses on the implementation of new technologies in non-formal learning settings, like museums and archaeological sites.