Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: A case study

Article *in* International Journal of Mobile Learning and Organisation · July 2016

DOI: 10.1504/IJMLO.2016.077867

CITATIONS
53

READS
3,656

3 authors:

Stamatios J Papadakis
University of Crete
136 PUBLICATIONS **688** CITATIONS
SEE PROFILE

Michail Kalogiannakis
University of Crete
154 PUBLICATIONS **755** CITATIONS
SEE PROFILE

Nicholas Zaranis
University of Crete
60 PUBLICATIONS **465** CITATIONS
SEE PROFILE

Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study

Stamatios Papadakis*, Michail Kalogiannakis and Nicholas Zaranis

Department of Preschool Education,
Faculty of Education,
University of Crete,
Crete, Greece
Email: stpapadakis@gmail.com
Email: mkalogian@edc.uoc.gr
Email: nzaranis@edc.uoc.gr
*Corresponding author

Abstract: In recent years, the teaching of programming and development of fundamental programming concepts at the preschool age has attracted the interest of the educational and scientific community. International research has highlighted that teaching programming to young children has a crucial influence on the development of their cognitive functions. There are currently plenty of available programming environments suited for preschoolers. Researchers are adapting their views concerning the age threshold at which young children can effectively get involved with programming. A new programming environment, which was designed to help preschoolers familiarise with basic programming concepts, in a developmentally appropriate manner, is ScratchJr. This study performs a brief introduction to the characteristics of ScratchJr as well as a presentation of the results of a small-scale pilot study for the evaluation of ScratchJr as means of teaching basic programming concepts in the preschool classroom.

Keywords: ScratchJr; preschool education; basic programming concepts; computational thinking.

Reference to this paper should be made as follows: Papadakis, S., Kalogiannakis, M. and Zaranis, N. (2016) 'Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study', *Int. J. Mobile Learning and Organisation*, Vol. 10, No. 3, pp.187–202.

Biographical notes: Stamatios Papadakis is a graduate of the Economics and Business (AUEB) University, Athens, Greece, Department of Information. He received a MSc in Education from the University of the Aegean, Greece, and a PhD from the University of Crete, School of Education. He has been working for a series of years as an ICT teacher in public sector Secondary Education. He has published many articles in journals and has presented several papers in conferences. His research interests include ICT in education, mobile learning, novice programming environments and teaching of programming in primary and secondary education.

Michail Kalogiannakis is an Assistant Professor in the Department of Preschool Education at the University of Crete and Associate Tutor at School of Humanities at the Hellenic Open University. He has graduated from the Physics Department of the University of Crete and continued his post-graduate studies at the University Paris 7-Denis Diderot (D.E.A. in Didactic of Physics), University Paris 5-René Descartes-Sorbonne (D.E.A. in Science Education) and received his PhD degree at the University Paris 5-René Descartes-Sorbonne (PhD in Science Education). His research interests include science education in the early childhood, science teaching and learning, e-learning, the use of ICT in education, distant and adult education. He has published many articles in international conferences and journals and has served on the program committees of numerous international conferences.

Nicholas Zaranis graduated from the Department of Electrical and Computer Engineering of the NTUA. He took the MSc in ICT from the University of Essex. Also, he received his Master of Science and his PhD from the University of Athens. He has worked for a series of years in the private sector as Electrical Engineer in Hewlett Packard Hellas and in public sector as teacher in Secondary Education. He is currently Associate Professor in the Department of Preschool Education at the University of Crete. His research interests include ICT in education, educational software, and teaching of Mathematics using ICT.

1 Introduction

Even today, school education sometimes seems trapped in following the path of the 19th-century learning logic (teaching and examination). This is opposed to the modern 'road to knowledge', which concentrates on building foundations for the gradual conquest of abstract concepts as means of leading children to 'learn how to learn'. Apart from that, the current generation of 'digital natives' is growing in a digitised world in which technology is evolving rapidly, creating new fields of study, new forms of employment, requiring new skills and abilities (Yang et al., 2015). Despite the fact of the initial concern about the use of technology in the education of young children (Cordes and Miller, 2000), proof of the benefits of using developmentally appropriate interactive technology has been well documented (Couse and Chen, 2010). In recent years, the rapid development of technology has contributed to the development of new educational tools (Yin and Fitzgerald, 2015). Additionally, the vast dissemination of various forms of ICT (Information and Communications Technologies) has lead students to the gradual conquest of functional knowledge that facilitates the development of higher levels of practical skills useful in STEM education (science, technology, engineering, and math) that is applicable to real-life contexts. As Wing states, this knowledge, referred to as computational thinking, builds on the power and limits of computing processes, giving students the necessary methodology and models to solve problems and design systems. Today, computational thinking is such a fundamental skill for everyone that we should add it to every child's analytical ability along with reading, writing, and arithmetic (Wing, 2006).

In this context, many educational institutions and researchers worldwide have created numerous tools and programming environments (Chookaew et al., 2015), which even target at preschool age children. The reason for this is that numerous studies have confirmed the benefits generated by the teaching of programming concepts in the development of basic cognitive skills, which are associated, for example, with the mathematical ability and the development of logical thinking in children of preschool and early primary school age (Kazakoff and Bers, 2012; Kazakoff et al., 2013; Grover and Pea, 2013; Strawhacker et al., 2015b). This implementation can be more efficient when it is combined with the creation of technological tools that take advantage of powerful modern mobile devices (Yin and Fitzgerald, 2015).

Programming is traditionally associated with the development of the above-mentioned skills, as it requires the use of structured thinking (Portelance, 2015). Programming environments such as 'Tynker' (https://www.tynker.com/), 'Hopscotch' (http://www.gethopscotch.com/) and the 'Move the Turtle' (http://movetheturtle.com/) aspire to introduce children to programming in a manner, which is compatible with their level of cognitive development. At the same time, international organisations such as Code.org (http://code.org/) and Code Academy (http://www.codecademy.com/) are planning the learning of basic programming principles through interactive online courses that even include preschoolers (Portelance and Bers, 2015). Since 2013, the UK is the first country in the world to mandate computer programming in primary and secondary schools (Department for Education, 2013).

ScratchJr is a new programming environment, which provides a platform for the development of problem-solving skills in a playful way and can be ideal for the development of reading, writing, and arithmetic skills in preschool education. The creators of ScratchJr aim exactly to fill the gap created by the lack of a developmentally appropriate teaching and development platform. With ScratchJr young children learn fundamental programming principles and concepts while at the same time create their animated stories and games in a developmentally correct and playful way.

The present study deals with the importance of the development of fundamental programming concepts in preschool education and the necessity of creating the programming environment ScratchJr. It also presents a preliminary, small-scale pilot study indicating that the use of ScratchJr contributes significantly to the development of fundamental programming concepts and computational thinking in preschool education.

2 The importance of developing fundamental programming concepts and computational thinking

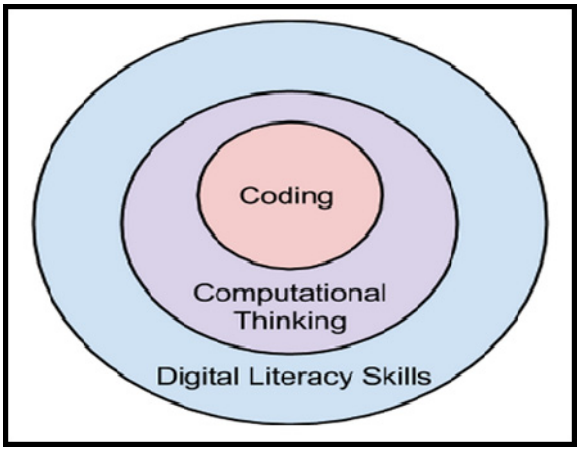
As early as 2006, in his book titled 'The World is Flat', Thomas Friedman stated that the economy needs 'Versatilists', namely people who are specialised not only in a professional field but also in Informatics. According to the same author, Computer Science is the connective tissue that enables 'Versatilists' to bridge their professional expertise with technological innovation (Seehorn et al., 2011).

Prior to this author, other researchers had also reported the necessity of understanding Computer Science, using the term computational thinking (CT). Why is CT so important? It is important because it allows one to solve problems, to design systems, and understand the potential and limitations of human intelligence and of machines. It is a skill that powers the modern world and, therefore, all students should possess and

develop skills in this area (UNESCO, 2005; Berry, 2013). CT is a skill today's students need to be taught, in order to adequately prepare for the workplace but also to be able to participate effectively in the modern digital world. CT is defined as a problem-solving process. It includes the ability to use knowledge for the exploitation of various tools of information technology, the steps-actions needed to solve a problem and logical organisation and data analysis (UNESCO, 2005; Pougatchev, 2007). In education, CT as a problem-solving methodology can be automated and be used across the spectrum of the curriculum (Barr and Stephenson, 2011). Following this approach, CT is allowing the combined use of Computer Science in all disciplines providing the means for analysing and developing solutions to all problems that can be solved computationally (Seehorn et al., 2011).

There is a growing recognition by the academic and scientific community that 'coding is the new literacy' for preschool education. This position implies the idea that CT, similar to reading and writing, may facilitate the development of other skills such as problem-solving and the development of the necessary for our time, 21st-century skills (Figure 1).

Figure 1 The relationship between coding, computational thinking and digital literacy skills (see online version for colours)



Source: Google for Education (2015)

The term computational thinking is not new. Seymour Papert first used it to describe not just the procedures and concepts used for the solution of problems and the design of computer systems, but also the application of the above in the perception and comprehension of natural phenomena (Papert, 1996). The term, however, is defined in depth by Jeannette Wing, which describes computational thinking as a rich set of analytical methods that effectively involve the human and the machine element in the solution of various problems (Wing, 2006). These methods include specific tasks such as programming, testing and debugging, and the use of abstract concepts such as data representation.

2.1 The importance of development of fundamental programming concepts and computational thinking in preschool education

In the first years of the introduction of computers in school classes, there was an intense debate whether the use of technology in preschool and primary education was a suitable educational development (Clements and Sarama, 2005). However, today the initial doubts and aphorisms have been eliminated, resulting in the introduction of ICT even as early as in preschool education. Compared to traditional instruction or information from textbooks, learning with the use of ICT seems to be a more attractive way of learning that can trigger the interest and motivation of preschoolers (Hwang and Chang, 2011).

Moreover, even the teaching of programming is now considered acceptable at early ages. Considering that 'coding is the new literacy', the teaching of programming and the use of corresponding languages and programming environments has gained considerable popularity in recent years in western countries. The major goal of programming courses is to help students learn to solve problems with program design rather than merely memorising the syntax of the programming language and the operations of the programming tools (Wang et al., 2015). As a result, in the USA, for example, federal education programs and private initiatives, such as the non-profit organisation Code.org (www.code.org), have made the teaching of Computer Science and technology literacy acquisition a priority for young students (Portelance, 2015; Strawhacker et al., 2015a).

As early as September 2014, ICT is out in primary schools of Great Britain. It has been replaced by a new 'computing' curriculum including coding lessons for children as young as five. Children aged 5–7 need to know the use of simple commands and to predict the behaviour of simple programs, while children aged 7–11 should be aware of how to apply repetition, selection and the use of variables (European Schoolnet, 2015). As characteristically mentioned in the National Curriculum of Great Britain, '*A high-quality computing education equips pupils to use Computational Thinking and creativity to understand and change the world*' (Department for Education, 2013).

Nevertheless, though it may seem strange, teaching programming to children is nothing new. On the contrary, it has its roots in the 1970s and 1980s, since the most notable, perhaps, initiatives of MIT professor Seymour Papert (Barseghian, 2013). For Seymour Papert, who introduced the educational use of the Logo language, coding was more than a matter of just teaching a subject or another programming language. He believed that it was a powerful tool for students to develop their cognitive skills. As he points out: '*I began to notice that children who had learned to program computers could use very specific computational models in their personal way of thinking and learning and therefore to be better prepared for their future academic and professional development*' (Papert, 1993, p.21).

Additional studies have shown that when children learn through well-target-staged program building and/or debugging, they are motivated to get actively involved and not just participate in the processing of an activity. They are led to a better comprehension of mental objects not only in the programming field but also objects that pertain across the curriculum (Brennan, 2011; Barseghian, 2013). Even preschoolers can create and study third party ready-made programs in a way that demonstrates a deep understanding of the basic programming concepts. Specifically, they can name actions corresponding to instructions, classify events in a logical order, and even create a simple program to achieve a hypothetical goal (Brennan, 2011; Barseghian, 2013).

3 A short introduction to ScratchJr

Various studies have shown that children as young as four years old can understand basic programming concepts and can create and program simple robotic constructions (Bers and Horn, 2010). Moreover, studies using the Logo programming language showed that when the teaching of programming is introduced in a structured way, it can help young children develop a variety of cognitive skills (Clements, 1999). However, the same studies reveal the inadequate design of programming environments aimed for use by young children. The strict syntax of the text-based programming languages such as Logo can finally discourage young children. Alternatively, graphics programming environments can potentially simplify the syntax difficulties, but at the same time frequent use of text in such cases poses an additional challenge for children of this age (Clements, 1999).

The creation of ScratchJr was based exactly on the lack of a developmentally appropriate environment for creating digital stories and learning about basic programming concepts in preschool education. Although there are a plethora of remarkable programming tools such as Hopscotch, Kodable, Run Marco, Code Studio, CS-First, Tynker, Daisy the Dinosaur, Cargo-Bot, Hakitzu Elite, Lightbot, Move The Turtle, they are in their entirety aimed at children aged at least 7 or 8 years old. In contrast, the plethora of self-proclaimed educational applications (mobile or not) and individual technologies which are addressed to preschoolers focuses on the development of basic skills such as letter and number identification, rather than the creation of content and the development of higher level skills (Zaranis et al., 2013; Papadakis et al., 2016b). The aim of ScratchJr creators is to give preschoolers a programming environment whereby young children in a developmentally correct and playful way learn fundamental programming principles and concepts while creating their animated stories and games. As one can read in the official website of ScratchJr (www.ScratchJr.org), the creators of the environment aspire '*children not only to simply learn to write code but to also encode their learning*' with its use (ScratchJr.org, 2015a). ScratchJr (Scratch Junior) is an introductory programming environment that allows young children (5–7 years) to 'discover' the basic programming concepts by creating projects in the form of interactive stories and games. ScratchJr and its accompanied mobile applications for Android and iOS ecosystems was created jointly by the Developmental Technologies Research Group and Media Lab's Lifelong Kindergarten Group at Tufts University and Massachusetts Institute of Technology (MIT) and the privately funded company Playful Invention.

The template for creating ScratchJr was another research product of the MIT Lifelong Kindergarten Group, the popular programming environment Scratch (Flannery et al., 2013). Scratch (http://scratch.mit.edu) is a visualised programming environment that enables students to develop animations or games by just dragging and dropping icons representing particular programming instructions (Wang et al., 2015). Additionally, ScratchJr takes advantage of the popularity of mobile devices with young children (Zaranis et al., 2013; Papadakis et al., 2016a) as it is available both for smart mobile devices with iOS or Android operating systems and screen sizes up to 7 inches. The disposal of the personal computer version is envisaged for the forthcoming future. During ScratchJr design phase, the researchers removed many of the characteristics of Scratch in order for the environment to be developmentally appropriate for preschoolers (Resnick et al., 2009). Consequently, at a technical level, ScratchJr has certain limitations compared to Scratch. One of them, for instance, is the lack of variables. However, the environment

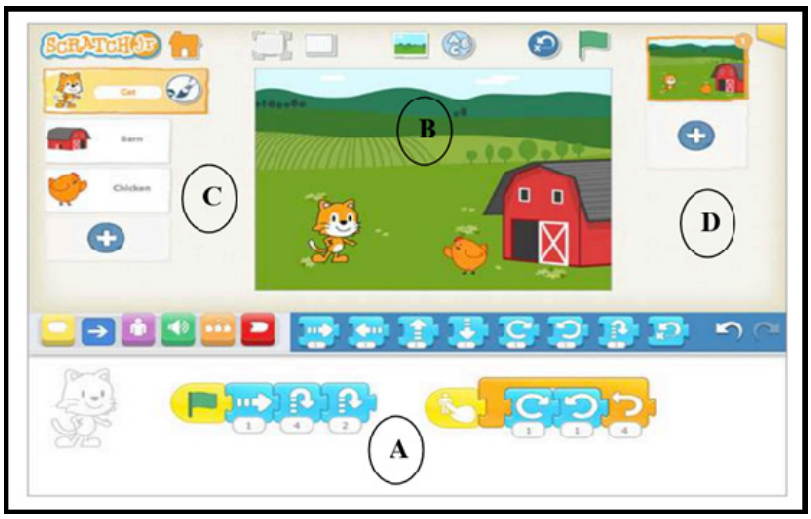
has adopted some 'advanced' features of Scratch, such as the use of the 'broadcast' block. In summary, ScratchJr, despite being much simpler than Scratch, still includes several powerful options that allow preschoolers to express themselves creatively.

A graphical programming language that consists of 28 different blocks is the core of ScratchJr. Users create their projects by connecting blocks at reasonable sequences enabling the characters that appear on the screen to move, change their appearance and/or produce sounds. With a user-friendly interface, preschoolers can further customise their project by adding multiple pages, integrating various characters and/or backgrounds using application libraries or external sources. Alternatively, they can create their characters or backgrounds using the embedded paint application: add text, set up their own sounds, etc. The fact that ScratchJr is addressed to young children does not limit the ability of preschoolers to include diversification and complexity in the produced projects.

The interface for creating a project with ScratchJr consists of four parts (Figure 2):

- A: the command editor or programming area where the user connects programming blocks to create scripts, instructing the character what to do, etc.
- B: the scene or stage in which the characters 'act' following orders-instructions.
- C: a list of the characters that have been added to the stage.
- D: a collection (gallery) ranging from one to four pages, each of which is associated with a new scene and working environment for the continuation of the project (ScratchJr.org, 2015b).

Figure 2 A snapshot from ScratchJr environment



When the application starts up, it opens up in an active project condition in which there is a character (cat) already on stage. The command tiles associated with the character's movement are immediately visible and ready to be used (Figure 3).

Figure 3 ScratchJr interface when starting a new project

In order to create a simple program, the user can simply choose any of the eight motion blocks in the script area just by tapping, with a drag and drop method. Similar to Scratch, preschoolers are encouraged to create more complex projects including several blocks, as blocks are joined easily with each other like pieces of a jigsaw puzzle. Syntax errors are impossible to occur in the ScratchJr environment because blocks are in such a way designed to allow only the logical connections between them (Figure 4).

Figure 4 An example of a completed project in ScratchJr

Additionally, the designers of ScratchJr changed its vertical orientation to landscape in an attempt to simulate the writing process when preschoolers create scenarios. They partially redesigned its interface to become more developmentally appropriate for

children of preschool age (Portelance and Bers, 2015). As it is well known, in early childhood, fine motor skills and visual motor integration for efficient hand–eye coordination, essential for mouse or touchpad control, are not highly developed and, therefore, can hinder the effective use of the software (Dankert et al., 2003). The various components of the ScratchJr environment are quite large. This facilitates the targeting of both blocks and buttons when using the mouse cursor either on a computer or with a fingertip on a tablet. Additionally, all components of ScratchJr are identified by the use of easily noticeable icons so that preschoolers who lack the ability to read, learn quickly and easily the functions of the learning environment. Even in contrast with Scratch, the layout of ScratchJr was redesigned to provide the absolutely necessary tools, without the use of a complicated menu. In general, all of the environmental features are redesigned to minimise required mouse moves and/or user's fine motor challenges.

In terms of programming, ScratchJr offers the following features (Flannery et al., 2013):

- Low Floor – High Ceiling: it is easy for a preschooler to start programming with ScratchJr. However, at the same time the preschooler is provided with adequate 'space' to create projects that vary in complexity, keeping the tool developmentally appropriate for the mental and age range of the user.
- Wide Walls: ScratchJr enables multiple learning 'paths' and various forms of exploration and creativity.
- Tinkerability: it is easy for preschoolers to incrementally and gradually create projects and enhance their knowledge through experimentation with new ideas and features.
- Conviviality: ScratchJr Graphical User Interface (GUI) is considered user-friendly, improving children's holistic development, by encouraging investigation and promoting exploration.

4 Research description

The purpose of this small case study, which also serves as a pilot for a broader research, was to investigate the effect of using ScratchJr for teaching basic programming concepts and more generally for the development of computational thinking in preschool education. Computational thinking in preschool and the first two years of compulsory primary school education is typically considered as comprising of several key factors that aid in learning to algorithmically solve problems (Calderon et al., 2015). For this purpose, a teaching intervention was implemented in which preschoolers used ScratchJr for the creation of their projects. The ethical considerations and guidelines concerning the privacy of individuals and other relevant ethical aspects in social research were carefully taken into account throughout the whole research process. Requirements concerning information, informed consent, confidentiality and usage of data were carefully met, both orally and in writing, by informing the preschool staff, children, and guardians of the purpose of the study and their rights to refrain from participation.

For the evaluation of the level of basic programming concepts knowledge (and also computational thinking) preschoolers achieved at the end of the teaching intervention, several aspects of measurements were performed. More precisely preschoolers were

measured in their ability: (a) to understand a single block, (b) to transform individual blocks in an integrated operational program, (c) to create a complex project, and (d) to understand the blocks that make up a project.

The sample of the study consisted of 43 preschool children (22 boys, 21 girls) who were attending classes in a public and a private kindergarten in the region of Crete, Greece, during the school year 2014–2015. The data were analysed using IBM SPSS 23.0 Software, and the significance level adopted was 5% ($p < 0.05$). The Institutional Review Board (IRB) of the University of Crete reviewed the scientific merit of the research, the validity of the research strategy and the adherence to the accepted practices in the respective field of study for human subject research protocols.

The teaching intervention had a duration of 13 hours (Table 1). It was adapted from the ScratchJr 'Animated Genres' curriculum as described by Portelance and Bers (2015) and the activities included in the book entitled 'The Official ScratchJr Book' (Bers and Resnick, 2015). During the teaching intervention, preschoolers became familiar with the development environment, learned basic programming concepts (blocks) and created their projects. Specifically, the first 11 hours were devoted to tutoring by the researchers and the last two hours on the creation of projects by the children. The duration of each teaching intervention was hourly, while the frequency of sessions was twice weekly (or once every 3–4 days).

Table 1 Content of the teaching intervention

Teaching hour	*Modules*	*Learning objects*
1st 2nd 3rd 4th	An introduction to ScratchJr	Open the app, make, save a new project, use the green flag, add a background, add another character, add a title
5th 6th 7th	Animations	Make the character move, make the character turn, hide and seek activities, repeat forever
8th 9th	Stories	Use your voice, create a character, insert – change the page
10th 11th	Games	Make various games (e.g. pick a peach)
12th 13th	Project time!	Free choice project creation

The teaching strategy adopted a constructivist approach as preschoolers were urged to get actively involved in their process of learning (Kostelnik et al., 2007). As a result, in all stages of implementation, the open-ended exploration and the creation of various activities and projects were encouraged. Preschoolers implemented activities that were developmentally appropriate for their age such as sorting objects by size, shape, and colour, the logical completion of a series of actions, etc. (Kazakoff and Bers, 2012) (Figure 5). For the implementation of the teaching intervention tablets with Android Lollipop (5.0) mobile Operating System and a screen size of 10.1 inches were used.

Figure 5 A preschooler involved in a constructivist learning activity

There is a diversity of approaches to assess fundamental programming concepts as researchers worldwide have experimented with a variety of ways to measure the effectiveness of a development process. One common approach towards understanding the level of development of fundamental programming concepts is through the project analysis inspection of a student's use of a programming environment. Portelance (2015) states that the typical research approaches for secondary education students or novice programmers are comprised of the determination of the frequency with which students use different components of the programming environment in their projects. These include the use of control flow statements, objects, variables, loops, design patterns such as user interaction, and the use of concepts that requires a higher level of computational thinking as the abstraction, encapsulation, and inheritance.

However, the existence of assessments that are developmentally appropriate for preschoolers is seldom (Portelance, 2015). Those available, in their majority, assess the degree of development of young children fundamental programming concepts, focus on their ability to use and prioritise various commands, to recognise and place programming blocks in the correct order, often with the help of artefacts such as wooden or paper blocks (Portelance et al., 2015).

Owing to the young age of the sample, the researchers could not follow one of the typical procedures for the evaluation of projects developed in a programming environment. For that reason, they adapted and partially redesigned the procedures and methods that have been applied by Portelance and Bers (2015) as well as by Strawhacker et al. (2013), which evaluate the degree of development of programming concepts by preschoolers following different approaches. These researchers have proposed a basic framework for computational and digital skill evaluation in which the score is inversely proportional to the number of errors in the code of a program. A student's correct answer in a question was marked out of zero, whereas the score of a student increased according to the number of incorrect answers per question.

Specifically, to evaluate the ability to identify different commands, five different short stories, each of which included the activity of one or more characters, were presented to preschoolers by the researchers. The code of each story was not visible.

Subsequently, children were asked to circle on a page which, ScratchJr blocks, has been used for the creation of the story (program – project) that they had just watched. In another identification activity, preschoolers were also asked to assign commands (blocks) in five different half-baked projects of escalating difficulty (Lin, 2012).

5 Indicative results of the teaching intervention

As already mentioned, the present research is a small-scale preliminary pilot study. However, early qualitative and quantitative analysis of the data confirmed the ability of children to learn basic programming concepts even at preschool age. A statistical data analysis found that preschooler gender does not affect performance in computational and digital skills as the use of independent samples *t*-test showed a not statistically significant result, $t(41) = -.37$, $p > 0.05$. Therefore, there was no significant difference between boys ($M = 2.2$, SD = 1.09) and girls ($M = 2.1$, SD = 1.06) performances. Subsequently, an analysis of Pearson's correlation coefficient (r) showed that the age of children didn't affect their performance in understanding basic programming concepts, $r(41) = 0.07$, $p > 0.05$).

The most used blocks were Motion blocks, with 'Move Right' being the most frequently used block. The results are consistent with the earlier research of Portelance et al. (2015) which showed that kindergarten students dedicated most of their programming blocks to moving their characters around the screen. Notably, statistical analysis revealed there was no significant difference between the ratio of motion programming blocks between males and females $t(41) = -0.17$, $p > 0.05$. Our results suggest that boys ($M = 1.8$, SD = 0.90) and girls ($M = 1.7$, SD = 0.92) perform similarly to each other in various tasks.

Additionally, the qualitative data analysis showed that preschoolers encountered more difficulties and correspondingly made more mistakes when the scene involved more than one character. Most common mistakes were on activities in which the character was jumping on the stage. For example, whereas the character was using the block 'jump', preschoolers mistakenly thought that the blocks 'move up' and 'move down' had been used. Next most common were mistakes where preschoolers confused the usage of 'spin left' block with the 'spin right'. Those results are in correspondence with the results of the study of Strawhacker et al. (Forthcoming).

Moreover, findings reveal that ScratchJr enhances student interest by making the learning experience fun. Similarly, animated scenarios showed high levels of engagement among students. Specifically, ScratchJr allowed children to engage in deep reflection as they solved problems and collaborated with their peers, both of which activities enhanced their learning experience. The findings showed that students enjoyed playing with ScratchJr and indicated that this programming environment is efficient for the learning of some programming constructs in early childhood and lower level education. ScratchJr helped promote collaboration and problem-solving skills as children became involved in the development process of their projects.

6 Discussion – perspectives

In the 21st century, the demand for the development of computational and digital skills, as elements of the fundamental literacy, will become increasingly intense. Programming environments, such as ScratchJr, will appear to support the effort to fulfil this need. The present study aimed to examine whether the use of ScratchJr through the implementation of developmentally appropriate activities provides preschoolers with new learning opportunities for developing computational and digital skills. For the purpose of this study the activities that were utilised allowed preschoolers to familiarise themselves with and use computational and creative characteristics of ScratchJr which are appropriate for their age or developmental stage.

The results showed that ScratchJr is especially attractive to preschoolers due to their active engagement in game-based problem-solving activities. Preschoolers developed computational and digital skills (abilities to abstract, compartmentalise and synthesise) that made solving problems easier and helped them create animations, collages, stories, and games. They participated with undiminished intrinsic interest and pleasure in the targeted activities' actions, which presented an efficient way of increasing enthusiasm for problem-solving when using the ScratchJr programming environment.

The findings of this study confirm previous research results. It provided evidence that even preschoolers and kindergartners can learn to code. The teaching of programming, even as early as in preschool education, provides a unique environment in which preschoolers, in an enjoyable and meaningful way, explore and indulge in relatively abstract for their age concepts, such as logical reasoning and problem-solving skills. At the end of the intervention, the majority of preschoolers were able to observe a project and to conclude, through reverse logic, in the commands (blocks) they had to use to reconstruct the respective applications. Additionally, as computational thinking is a type of analytical thinking that shares many similarities with problem-solving, designing and evaluating processes, and systematic analysis (Bers, 2010) in preschool education, the use of ScratchJr has the potential to help children increase sequencing skills and develop various academic skills such as science process understanding and mathematical concept development. As a result, ScratchJr could be implemented in early childhood as a teaching tool, set up in a developmentally appropriate way: by integrating other disciplines, helping children develop cognitive, conceptual, language and collaborative skills (Toh et al., 2016). As Bers et al. (2014) state when given age-appropriate technologies, curriculum, and pedagogies, young children can actively engage in learning computer programming. Children can then take their first steps into developing computational thinking.

There are several limitations to this study that should be acknowledged. Owing to the chosen method of teaching intervention, not all preschoolers did receive the same training during the study. Many preschoolers were absent from school during the days of intervention. Even though the researchers made attempts for individualised teaching for infants who were absent, this form of teaching cannot be considered equivalent to the learning environment of the classroom. Additionally, the study was conducted in two different classes and, inevitably, the approach of the teaching staff in promoting social/behavioural skills differed. Additionally, preschool teachers were encouraged to contribute to classroom management and took part in ScratchJr activities according to their familiarity with digital skills; thus, preschoolers experienced the implementation of the activities differently.

Also, the narrow geographical focus is highlighted as one of the research weak points, as well as the small sample size and short duration of implementation of the teaching intervention. These shortcomings prevent, to some extent, the generalisation of the results beyond the cases that have been studied. Additionally, this research does not answer questions related to the development of computational thinking, such as the effectiveness of individual (solo) programming compared to pair (peer) programming, the sustainability of children's knowledge of basic programming concepts per time unit, etc. Many of these questions are expected to be answered in the near future during the implementation of a wider in aspects of sample and geographical focus research. Moreover, we plan to implement and standardise an 'instrument' that measures the achieved computational thinking and its development in preschool children.

References

Barr, V. and Stephenson, C. (2011) 'Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?', *ACM Inroads*, Vol. 2, No. 1, pp.48–54.

Barseghian, T. (2013) *Learn to Code, Code to Learn*. Available online at: http://ww2.kqed.org/ mindshift/ 2013/ 10/22/learn-to-code-code-to-learn/ (accessed on 6 January 2016).

Berry, M. (2013) *Computing in the National Curriculum: A Guide for Primary Teachers*, Computing at School, Bedford.

Bers, M.U. (2010) 'The TangibleK robotics program: applied computational thinking for young children', *Early Childhood Research & Practice*, Vol. 12, No. 2. Available online at: http:// ecrp. uiuc.edu/v12n2/bers.html

Bers, M.U., Flannery, L., Kazakoff, E.R. and Sullivan, A. (2014) 'Computational thinking and tinkering: exploration of an early childhood robotics curriculum', *Computers & Education*, Vol. 72, pp.145–157.

Bers, M.U. and Horn, M.S. (2010) 'Tangible programming in early childhood: revisiting developmental assumptions through new technologies', in Berson, I.R. and Berson, M.J. (Eds): *High-Tech Tots: Childhood in a Digital World*, Information Age Publishing, Greenwich, CT, pp.49–70.

Bers, M.U. and Resnick, M. (2015) *The Official ScratchJr Book*, No Starch Press, Inc., San Francisco, CA.

Brennan, K. (2011) *Creative Computing: A Design-Based Introduction to Computational Thinking*. Available online at: http://scratched.media.mit.edu/sites/default/files/CurriculumGuide-v20110923.pdf (accessed on 17 January 2016).

Calderon, A.C., Crick, T. and Tryfona, C. (2015) 'Developing computational thinking through pattern recognition in early years education', *Proceedings of the 2015 British HCI Conference*, ACM, New York, pp.259–260.

Chookaew, S., Wanichsan, D., Hwang, G.J. and Panjaburee, P. (2015) 'Effects of a personalized ubiquitous learning support system on university students' learning performance and attitudes in computer-programming courses', *International Journal of Mobile Learning and Organisation*, Vol. 9, No. 3, pp.240–257.

Clements, D.H. (1999) 'The future of educational computing research: the case of computer programming', *Information Technology in Childhood Education Annual*, Vol. 1, pp.147–179.

Clements, D.H. and Sarama, J. (2005) 'Young children and technology: what's appropriate', *Technology-Supported Mathematics Learning Environments*, Vol. 1, pp.51–73.

Cordes, C. and Miller, E. (2000) *Fool's Gold: A Critical Look at Computers in Childhood*, Alliance for Childhood, College Park, MA.

Couse, L. and Chen, D. (2010) 'A tablet computer for young children? Exploring viability for early childhood education', *Journals of Research on Technology Education*, Vol. 43, No. 1, pp.75–98.

Dankert, H.L., Davies, P.L. and Gavin, W.J. (2003) 'Occupational therapy effects on visual-motor skills in preschool children', *American Journal of Occupational Therapy*, Vol. 57, No. 5, pp.542–549.

Department for Education (2013) *The National Curriculum in England: Framework Document*, The Stationery Office, London.

European Schoolnet (2015) *Creative Use of Tablets in Schools*. Available online at: http://goo.gl/jteC9K (accessed on 15 April 2016).

Flannery, L-P., Kazakoff, E-R., Bontá, P., Silverman, B., Bers, M-U. and Resnick, M. (2013) 'Designing ScratchJr: support for early childhood learning through computer programming', *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*, ACM, New York, USA, pp.1–10.

Google for Education (2015) *Should My Kid Learn to Code?* Available online at: http://googleforeducation.blogspot.gr/2015/07/should-my-kid-learn-to-code.html (accessed on 18 January 2016).

Grover, S. and Pea, R. (2013) 'Computational thinking in K-12: a review of the state of the field', *Educational Researcher*, Vol. 42, No. 1, pp.38–43.

Hwang, G.J. and Chang, H.F. (2011) 'A formative assessment-based mobile learning approach to improving the learning attitudes and achievements of students', *Computers and Education*, Vol. 56, No. 4, pp.1023–1031.

Kazakoff, E. and Bers, M. (2012) 'Programming in a robotics context in the kindergarten classroom: the impact on sequencing skills', *Journal of Educational Multimedia and Hypermedia*, Vol. 21, No. 4, pp.371–391.

Kazakoff, E., Sullivan, A. and Bers, M.U. (2013) 'The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood', *Early Childhood Education Journal*, Vol. 41, No. 4, pp.245–255.

Kostelnik, M.J., Soderman, A.K. and Whiren, A.P. (2007) *Developmentally Appropriate Curriculum: Best Practices in Early Childhood Education*, Prentice Hall, New York.

Lin, C.L. (2012) 'Effects of sex differences and problem-solving strategies on digital game performance', *Journalism and Mass Communication*, Vol. 2, No. 9, pp.901–910.

Papadakis, S., Kalogiannakis, M., Orfanakis, V. and Zaranis, N. (2016a) 'Using Scratch and App inventor for teaching introductory programming in secondary education: a case study', *International Journal of Technology Enhanced Learning* (under publication).

Papadakis, S., Kalogiannakis, M. and Zaranis, N. (2016b) 'Improving mathematics teaching in kindergarten with realistic mathematical education', *Early Childhood Education Journal*, pp.1–10, doi:10.1007/s10643-015-0768-4.

Papert, S. (1993) *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York.

Papert, S. (1996) 'An exploration in the space of mathematics educations', *International Journal of Computers for Mathematical Learning*, Vol. 1, No. 1, pp.95–123.

Portelance, D.J. (2015) *Code and Tell: An Exploration of Peer Interviews and Computational Thinking With ScratchJr in the Early Childhood Classroom*, Master's Thesis, Tufts University, Boston, MA, USA.

Portelance, D-J. and Bers, M-U. (2015) 'Code and tell: assessing young children's learning of computational thinking using peer video interviews with ScratchJr', *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*, ACM, Boston, MA, USA.

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

Pougatchev, V. (2007) 'ICT-based education: Caribbean region perspectives', *Advanced Technology for Learning*, Vol. 4, No. 3, pp.132–139.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009) 'Scratch: programming for all', *Communications of the ACM*, Vol. 52, No. 11, pp.60–67.

ScratchJr.org (2015a) *Coding for Young Children*. Available online at: http://www.scratchjr.org/ (accessed on 25 January 2016).

ScratchJr.org (2015b) *About ScratchJr*. Available online at: http://www.scratchjr.org/about.html (accessed on 25 January 2016).

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cuniff, D., Boucher Owens, B., Stephenson, C. and Verno, A. (2011) *CSTA K-12 Computer Science Standards*, Revised 2011, CSTA Standards Task Force, CSTA, New York.

Strawhacker, A., Lee, M., Caine, C. and Bers, M-U. (2015a) 'ScratchJr demo: a coding language for kindergarten', *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*, ACM, Boston, MA, USA.

Strawhacker, A., Portelance, D. and Bers, M.U. (Forthcoming) 'What they learn when they learn coding: a study using the ScratchJr solve it programming assessment for young children', *Educational Technology Research and Development*.

Strawhacker, A., Portelance, D., Lee, M. and Bers, M.U. (2015b) 'Designing tools for developing minds: the role of child development in educational technology', *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*, ACM, Boston, MA, USA.

Strawhacker, A., Sullivan, A. and Bers, M.U. (2013) 'TUI, GUI, HUI: is a bimodal interface truly worth the sum of its parts?', *Proceedings of the 12th International Conference on Interaction Design and Children*, ACM, New York, pp.309–312.

Toh, L.P.E., Causo, A., Tzuo, P.W., Chen, I.M. and Yeo, S.H. (2016) 'A review on the use of robots in education and young children', *Educational Technology & Society*, Vol. 19, No. 2, pp.148–163.

UNESCO (United Nations Educational, Scientific and Cultural Organization) (2005) *Information and Communication Technology in Schools: A Handbook for Teachers or How ICT can Create New, Open Learning Environments*. Available online at: http://goo.gl/6aqhrT (accessed on 20 April 2016).

Wang, H.Y., Huang, I. and Hwang, G.J. (2015) 'Comparison of the effects of project-based computer programming activities between mathematics-gifted students and average students', *Journal of Computers in Education*, Vol. 3, pp.1–13.

Wing, J.M. (2006) 'Computational thinking', *Communications of the ACM*, Vol. 49, No. 3, pp.33–35.

Yang, T.C., Hwang, G.J., Yang, S.J. and Hwang, G.H. (2015) 'A two-tier test-based approach to Improving students' computer-programming skills in a web-based learning environment', *Education Technology & Society*, Vol. 18, No. 1, pp.198–210.

Yin, K.Y. and Fitzgerald, R. (2015) 'Pocket learning: a new mobile learning approach for distance learners', *International Journal of Mobile Learning and Organisation*, Vol. 9, No. 3, pp.271–283.

Zaranis, N., Kalogiannakis, M. and Papadakis, S. (2013) 'Using mobile devices for teaching realistic mathematics in kindergarten education', *Creative Education (Special Issue in Preschool Education)*, Vol. 4, No. 7A, pp.1–10.