Chapter

Evaluating a Course for Teaching Advanced Programming Concepts with Scratch to Preservice Kindergarten Teachers: A Case Study in Greece

Stamatios Papadakis and Michail Kalogiannakis

Abstract

Coding is a new literacy for the twenty-first century, and as a literacy, coding enables new ways of thinking and new ways of communicating and expressing ideas, as well as new ways of civic participation. A growing number of countries, in Europe and beyond, have established clear policies and frameworks for introducing computational thinking (CT) and computer programming to young children. In this chapter, we discuss a game-based approach to coding education for preservice kindergarten teachers using Scratch. The aim of using Scratch was to excite students' interest and familiarize them with the basics of programming in an open-ended, project-based, and personally meaningful environment for a semester course in the Department of Preschool Education in the University of Crete. For 13 weeks, students were introduced to the main Scratch concepts and, afterward, were asked to prepare their projects. For the projects, they were required to design their own interactive stories to teach certain concepts about mathematics or physical science to preschool-age students. The results we obtained were more satisfactory than expected and, in some regards, encouraging if one considers the fact that the research participants had no prior experiences with computational thinking.

Keywords: Scratch, preservice kindergarten teachers, programming, computational thinking

1. Introduction

According to the twenty-first century skills framework, digital literacy is an important skill for students to develop so as the ability to encode and understand code is becoming more and more a fundamental skill to master to participate actively to our digital society and economy [14]. National and European policies acknowledge the need to equip all citizens with the necessary competences to use digital technologies critically and creatively [28, 38]. As Wing [47] states "to reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (p. 33). Hence, its integration throughout all educational levels, as well as the early ages, is considered valuable. Evidence shows that even children as young

as 4 years old can engage in core computational thinking skills, provided they work with a developmentally appropriate tool that supports such learning [21, 34, 42].

Yet, the introduction of computational thinking (CT) in compulsory education requires support measures to prepare teachers [9]. Teachers themselves often have no formal education in computing and cannot communicate to their students' enthusiasm or understanding about what happens inside a computer to make it work [46]. Many primary teachers are unlikely to have the appropriate skill set to teach this new technical subject [6, 22]. Ref. [5] highlights that one of the obstacles to incorporating CT activities into the early childhood classroom is that early childhood educators have had little or no experience with technology concepts and processes. If teachers are to help young children learn CT concepts as well as STEM subjects (science, technology, engineering, and mathematics), their professional development ought to help them to explore content and teaching methods [11, 29]. This is considered important as children's experiences of science even at primary school inform their decisions about studying science, which impacts on the supply of STEM professionals [24].

Therefore, there is a need for widespread professional development to support in-service and preservice teachers in gaining the necessary experience, technical skills, confidence, and understanding of suitable pedagogies to implement this new curriculum successfully [6]. For these reasons, CT and programming is taught in many parts of tertiary education that are not necessarily directly relevant to or focused on information technology or STEM. These faculties include pedagogical departments in which students have a first familiarity with CT and programming either for their direct educational use or to be able to produce interactive and multimedia learning materials [16]. Many researchers have already used Scratch at the university in introductory programming courses, and their experiences report on students' high motivation and sometimes also on higher performance [25].

The rest of the paper is structured as follows: in the next section, the advantages of choosing Scratch as an introductory programming environment are outlined; the second section presents the methodology of the Scratch course employed in this article; and the third section documents the results. The final section discusses the results obtained, outlining the limitations and recommendations for future research.

2. The advantages of visual programming: Scratch

The inclusion of programming topics in the initial grades of school gives rise to debates about the best ways to teach these contents [17, 30, 32]. In recent years, new programming languages have been designed to be visually programmed without the need to learn the syntax, as it is the case with traditional languages [26].

Visual block-based programming environments are increasingly being used in introductory computer science lessons across elementary school grades. These environments, and the curricula that accompany them, are designed to be developmentally appropriate and engaging for younger learners [45]. Within these rich environments, the experience of coding can become playful and creative. They offer many opportunities for learning and personal growth, exploration, and mastery of new skills and ways of thinking [8]. Block programming eliminates the frustrations of syntax errors which afflict novice learning traditional computer programming languages [35]. Visual programming involves dragging and dropping instruction blocks together to form a program in a graphical development environment. The advantages of visual programming are [12]:

- Students do not need to learn syntax and cannot create syntax errors.

- Students can see what blocks (instructions) are available.
- Blocks often hide complex logic or operations in a single block.

The puzzle-like interface of these environments [10, 15, 33] allows novices to avoid syntax issues (e.g., semicolon use) and thus, allows them to focus on fundamental programmatic constructs (e.g., conditions, loops, variables). There is no typing error or misremembering of the syntax involved in the "bugs." The only possibility for an undesired outcome is the semantic error [43]. Since novices are not bullied by the compiler as they do not have to write codes following rigid syntactical rules, the programming is more meaningful and playful within Scratch [46]. This is a great relief for introductory programming and saves the learner much of the heartache traditionally forced on them by textual languages [46]. Given the large amount of software available and children-friendly programming environments such as Alice, Scratch, Greenfoot, and Kodu, teaching coding has become a more intuitive and engaging experience for young students [37] (see **Image 1**). In these graphical block-based programming environments, a novice programmer creates interactive applications by snapping together graphical pieces on the screen, like putting together a jigsaw puzzle. In addition, these environments are usually "low floor" and "high ceiling" and allow children to create their own complex computer programs, rich in sound and graphics [19].

On 15 May 2007, a revolutionary programming tool inspired by Logo (constructivist learning) was made available to the public. Scratch (https://scratch.mit.edu) is a free visual-based programming language environment especially developed for children and novices by the Lifelong Kindergarten Group of the MIT Media Lab. Like other visual block-based programming environments, Scratch presents a user-friendly visual language that encourages active methods, with a project-based learning and a role focused on student activity (see **Image 2**). Those

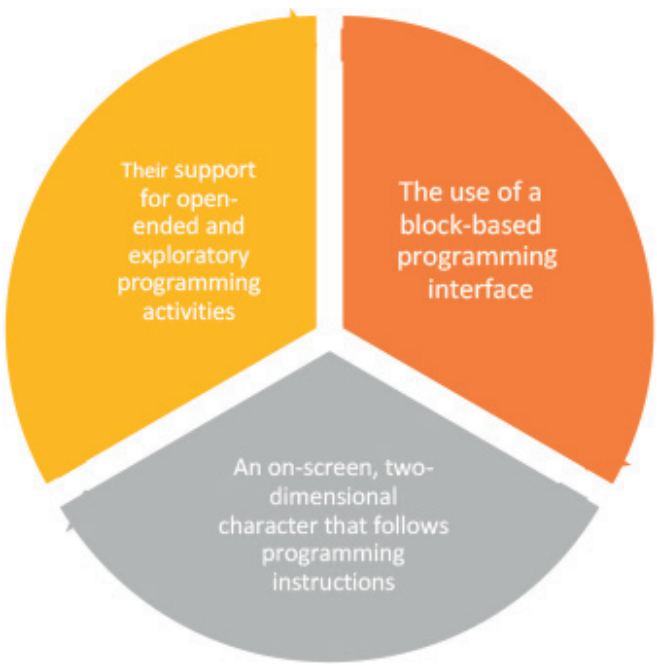


Image 1.
Key features of visual block-based programming environment (Adapted from [45]).

characteristics consist Scratch as one of the most popular tools used for introducing students to programming or better to CT (Evangelopoulou & Xinogalos, 2018). Scratch is designed to support children and novice learning through the process of experimenting and tinkering as it encourages learners to engage in creative learning experiences and express their ideas using code [44] enabling them to think creatively, reason systematically, and work collaboratively; all of which are essential skills required for the twenty-first century [20, 36].

Scratch can be used to program interactive stories, games, animations, music, and art [27, 31]. Those creations are called projects. A project is made up of sprites, which contain scripts, and they act on a stage [39]. The environment offers an online and an offline editor and an online community with millions of users sharing and remixing projects (Evangelopoulou & Xinogalos, 2018; [10]).

As Scratch has been developed with the aim of being very easy to use by anyone, regardless of age, background, or interests, it is being used by young people in schools, homes, and other learning environments around the world [44]. Only in August 2018, the Scratch website had almost 19 million visits with 115 million pageviews and 9 million unique followers! Also, Scratch is used at all levels of education across diverse fields, such as computer science, math, language, arts, social studies, and interdisciplinary projects (Evangelopoulou & Xinogalos, 2018; [10]). Even though it is claimed that Scratch appeals more to younger audiences [41], some universities (like Harvard, Berkley, and the University of California) have used Scratch as an introduction to programming [25, 43].

The next stage in the Scratch story is version 3.0. The beta version was released at https://beta.scratch.mit.edu/ on the first of August, and the official version will be available on January 2, 2019 [40] (see **Image 3**). Scratch 3.0 is written in HTML5. This means that with Scratch 3.0, the programmers will be able to play Scratch projects on their phone, create Scratch projects on their tablet, and control Scratch projects with their voice. There is also a version for kids for smart mobile devices, called ScratchJr (Scratch Junior) [10, 21, 22, 34, 42].

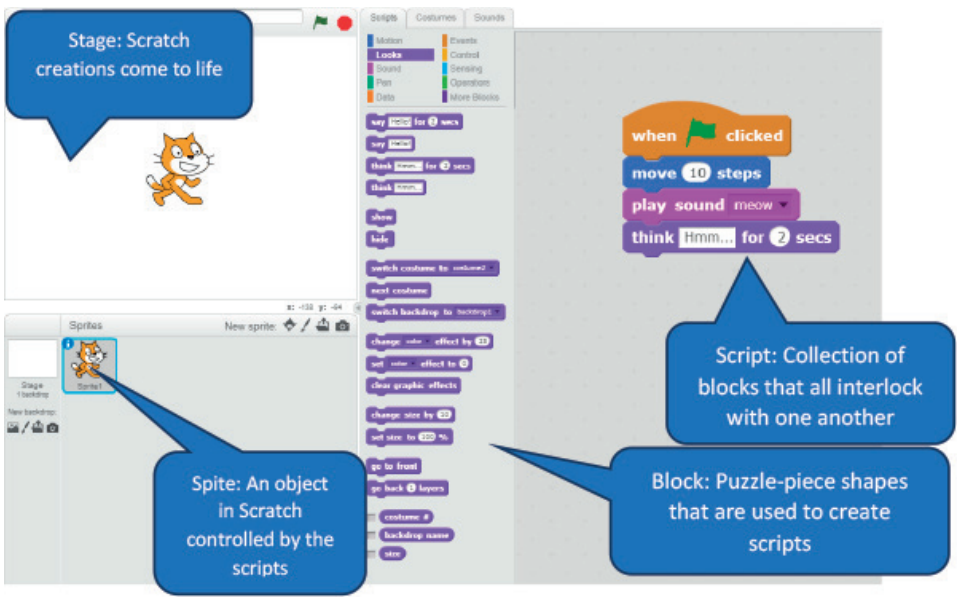


Image 2.
Scratch 2 layout.

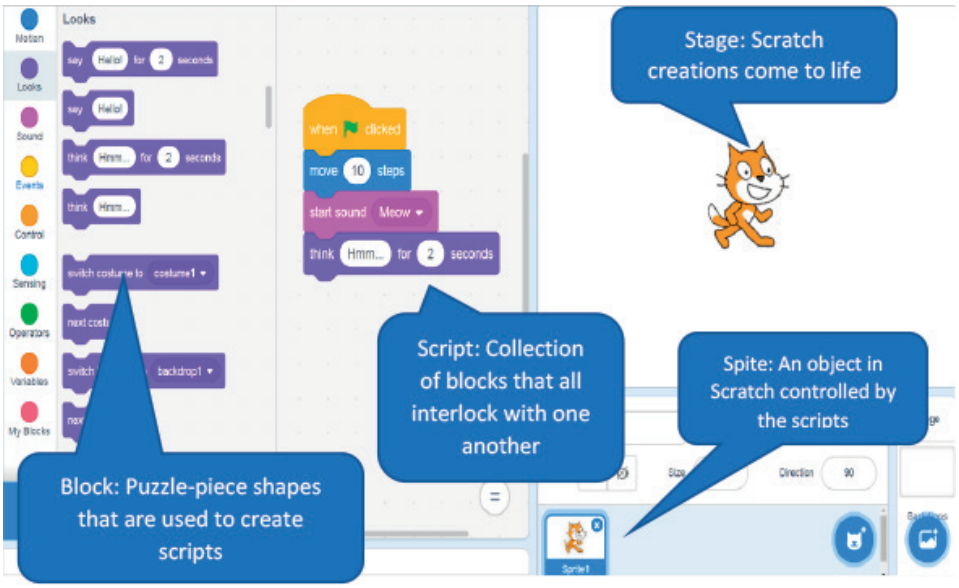


Image 3.
Scratch 3.0 layout.

3. Description of the course

3.1 The choice of the programming language

Novice programmers who are not interested in traditional approaches to coding become motivated when coding activities are introduced as a way to tell a story, or in connection with other disciplines and interest areas, such as music and art [44]. One of the main issues in the realization of the workshop was the choice of the programming language and how much time to allocate to the programming part [1]. As it is desirable that the preservice teachers be exposed to CT, and to its related concepts so as to be able to apply them effectively in the classroom and in learning activities, we decided to adopt Scratch as the introductory programming language environment at the Department of Preschool Education in the University of Crete. The reasons behind conception and design of this project are: we supposed that preservice teachers had different programming backgrounds and/or experience, and we felt that using Scratch as an introduction could be motivating, as it provides novices in programming with a meaningful and playful learning environment to create interactive games, animated stories, and simulations.

3.2 Objectives of the course

Technology and digital tools have become ubiquitous, but they can be ineffective or distracting if they are not integrated into the learning process in meaningful ways [5]. This paper presents an innovative approach that is guided by the constructionist philosophy developed by Seymour Papert. In constructionist learning environments, new knowledge is built through the programs created by learners [45]. In those environments not only can novice programmers design, build, and program their own interactive artifacts while having fun, but they can also learn how to work in groups and develop socioemotional skills [7]. In the process, they encounter powerful ideas from the realms of math, science, technology, and engineering [7].

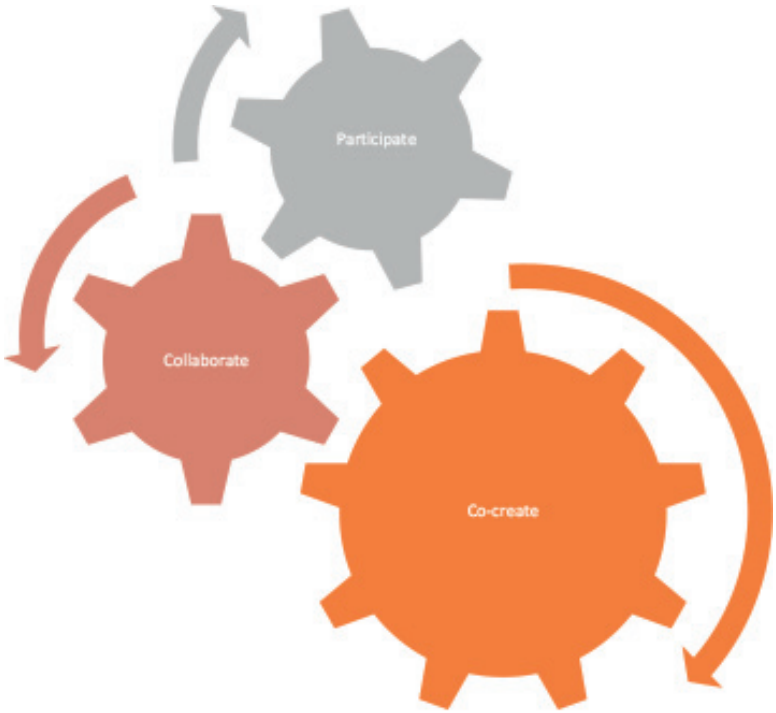


Image 4.
The trajectory of the course approach.

The course was developed to help preservice teachers introduce CS as a new subject to their students. It was also developed to demonstrate that even without a background or training in this subject, preservice kindergarten teachers have the ability to learn fundamental CS theory and concepts. It was focused on CS education in the context of developing higher-order thinking and problem-solving skills. We also wanted to encourage students to become innovative and think critically about how technology impacts their daily teaching techniques (see **Image 4**).

3.3 Course elements

Taking into account studies found in the literature [10, 43, 44], the course combines a little theoretical training with a strong practical component, encouraging the active participation of the trainee. Thus, the course elements are (see **Image 5**):

- Element 1. Scratch and applications built in Scratch. This element is divided into three parts:
  - The first one is about fundamentals and principles of CT.
  - The second part is about the Scratch environment, basic commands, control structures, and some advanced commands.
  - The third part is about the construction of projects in the form of animations, interactive stories, and educational games in Scratch. The Creative Computing Curriculum Guide (http://scratched.gse.harvard.edu/guide/) and the Scratch cards (https://scratch.mit.edu/info/cards/), a set of 12 cards which are available to download free from the Scratch website, were used as learning material, in order to help the students' teams to explore the features of Scratch on their own learning rhythm.

- Element 2. This element consists of building applications in the Scratch environment.

  - The students firstly were required to make their own version of the popular Angry Birds game. The idea was based on a similar project in the book entitled *Raspberry Pi Projects for Kids* [3]. In this game the player launches a bird through the air using a slingshot and attempts to hit all of the pigs at the other end of the level. This was a complex programming activity. In terms of the computational thinking framework, it involves the computational concepts of operators such as variables, control structures, keyboard-handling blocks, etc. The students had also to handle physics issues such as flight, gravity, and bouncing. For that reason, supplementary learning materials such as worksheets and group activity instructions were given to them. Furthermore, the educator advised the students how to manage the process of game development, working collaboratively, etc. Also, the educator offered his guidance to the students, helping them to complete their games and introducing even more complex CS concepts when needed.
  - Secondly, they were asked to create either an interactive story (based on an Aesop myth) or an educational game (trying to teach Greek language learning, math, or science). At the end of the semester, the students presented their projects. The program that each student created was also collected and analyzed to understand the outcomes of students' computational practices and their application of computational concepts.

3.4 Method

3.4.1 Participants

During the period between September 2017 and January 2018, 15 third-year female preservice kindergarten education students enrolled in a science education course entitled "Science education in early childhood" at a Greek university for

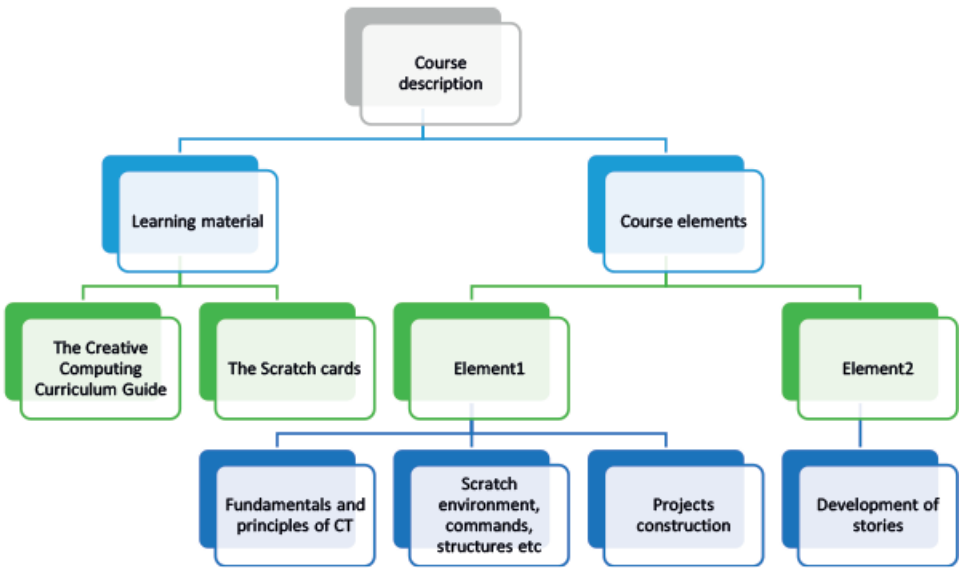


Image 5.
Course description.

13 weeks. The lessons were 3 hours per week. The course was offered as optional, and the students took part in the study after ethics approvals were received and all participants signed consent forms. All research participants had basic computer skills, but they had no previous experience with neither computational thinking nor the use of Scratch or any other programming environment.

3.4.2 Evaluation

In order to evaluate the course, we examined both cognitive (how effectively they learned) and affective (how enjoyable the experience was, and how motivated by it the students were) factors. Thus, in this study we collected both quantitative and qualitative data:

- The quantitative part was conducted in pretest/posttest quasi-experimental design. Moreover, to understand the learning of programming topics, we evaluated students' projects in terms of students' use of the elements of Scratch language as well as the project functionality and appearance. For that reason, students' project(s) were examined by using the Dr. Scratch tool.

- The qualitative approach used a short questionnaire and semi-structured interviews. Data were recorded through field notes, made by the researchers. This approach aimed at evaluating essentially three points:

  - The conception about the potential of Scratch and CT activities as a learning support tool
  - The intention to introduce a CT curriculum
  - The level of satisfaction about the course

The respondents were asked to answer both to closed questions (yes/not) and open questions ("Do you think that Scratch and coding activities can be a useful support learning tool? Why?," "Do you think about introducing some coding activities in your lessons? Why?").

4. Results

In this section we present and analyze the course results in terms of students' performance and satisfaction.

4.1 Performance analyses

Dr. Scratch is an online tool (http://drscratch.org/) that assesses Scratch projects with respect to seven "dimensions," namely, logical thinking (LT), data-information representation (IR), user interactivity (IN), flow control (FC), abstraction (AB) and problem decomposition, parallelism (PA), and synchronization (SN). A project can be graded (from 0 to 3) for each dimension in one of the levels, depending on the level of sophistication achieved by the project code [25, 26]. Thus, a total evaluation ranges from 0 to 21 (7 dimensions multiplied by [0–3]). We analyzed 15 different projects. The projects gathered were scored with values ranging between 10 and 20 (see **Table 1**).

Similar to other studies [27], this study revealed challenges with respect to the use of concepts, such as the parallelism and synchronization. Also, very few applications made use of random numbers and logical expressions. On the contrary,

Statistical measure	Dimension of computational thinking						
	PA	*LT*	*FC*	*IN*	*IR*	*AB*	*SN*
Mean	1.88	1.54	2.16	1.81	1.68	0.72	1.84
Std. dev.	0.38	0.29	0.41	0.32	0.29	0.22	0.41
Minimum	0	0	1	1	1	1	1
Maximum	3	3	3	3	3	2	3

Table 1.
Project score given by Dr. Scratch.

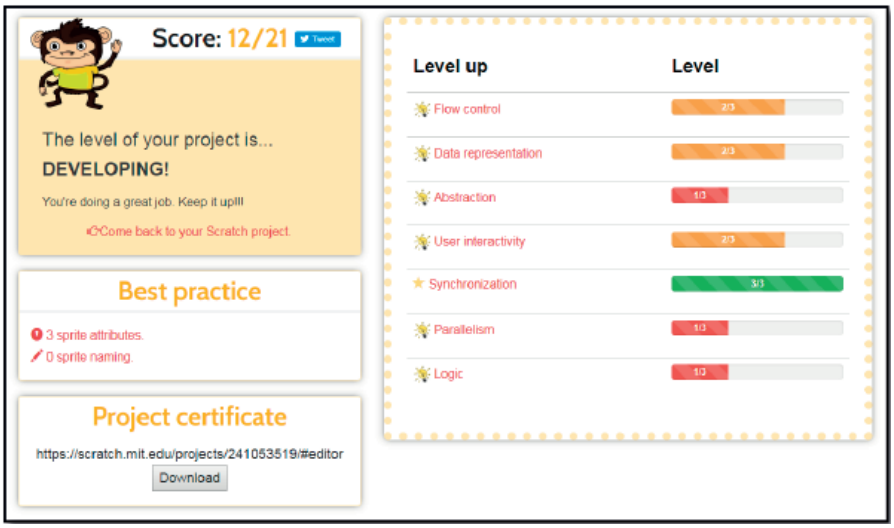


Image 6.
Example of Dr. Scratch project evaluation.

the frequently used coding concepts such as flow control and user interactivity reveal that the students in their projects make an adequate use of specific conditions and foresee users' interaction. Except the fact that Dr. Scratch provides feedback on several aspects which are related to computational thinking, the software categorizes the project developer skills in three different categories/levels: Basic, Developing, and Master. The 15% of the developed apps were "Basic," and 85% were "Developing." There were no projects on the "Master" level. **Image 6** shows an example of how Dr. Scratch categorizes developer skills. The screenshots of the graphical user interface, code parts, and Dr. Scratch scores of five randomly selected projects are displayed in the Appendix.

4.2 Students' self-efficacy analyses

To evaluate students' self-efficacy in utilizing programming and computational thinking within their future teaching endeavors, we adapted the Teachers' Self-Efficacy in Computational Thinking (TSECT) [4]. We used the first seven of the nine TSECT items (see **Table 2**). All items use a five-point Likert scale with options of strongly agree, agree, neither agree nor disagree, disagree, and strongly disagree. TSECT was given as a pre- and posttest before and after the intervention. Questionnaire analysis was performed with SPSS 23.

Item	Wording
1	I feel confident writing simple programs in Scratch
2	I know how to teach programming concepts with Scratch
3	I can encourage a positive attitude toward programming to my students
4	I can become a mentor teacher and support my students to use programming as a tool to explore other topics
5	I'm sure myself to use programming as an educational tool within a classroom
6	I can adapt methods, lesson plans, and curriculum materials for using programming as an educational tool
7	I can create lessons plans using programming as an educational tool

Table 2.
Modified TSECT instrument items.

A t-test of the pre and post-survey TSECT scale revealed a statistically significant increase in TSECT from pre (M = 12.03, SD = 4.39) to post (M = 18.14, SD = 3.59), $t(14) = 3.98$, $p < .0001$. From the students' answers, we can conclude that after the intervention, they feel themselves confident enough to create projects and they plan to incorporate programming as an instructional tool in their future classrooms.

Furthermore, after completion of the course, the researchers conducted a focus group interview with a structured interview form. All the students noted that the added cognitive effort was worthwhile and decided to bring coding activities into the early childhood classroom. The students noted that they experienced a significant shift in mindset during the course. Before the course started, all students identified the lack of CT knowledge and skills as a major challenge. After the course, all of them could successfully define key CT concepts. They expressed a high degree of confidence that they taught the CT lessons effectively contributed to their learning. Moreover, all students noted that they made major leaps in correcting their misconceptions about what CT is and understanding fundamental CT concepts. After the focus group interview, the researchers noted that the majority of the students could explain what CT is and describe the main concepts covered during the course. It is also worth to mention that all students indicated that they would like to continue CT training in the following academic year, if that was possible. They also mentioned that they would recommend the course to other students.

4.3 Limitations

In this chapter, we studied how a course helped preservice teachers to learn and introduce CT concepts into their daily teaching practices as a new subject to their students. The programming and teaching behaviors that emerged still need to be validated through further studies. Furthermore, since the data was collected from female students from one university department, the findings should be applied to subjects from other disciplines with caution. Moreover, it may be useful to employ a mixed method approach that incorporates long-term practical research methods for a deeper investigation of factors affecting attitudes and intentions toward using Scratch in respect to the students' gender.

5. Discussion and conclusion

Discussions about the appropriateness of technology in early childhood are mostly put aside, and the pressing question is not "Should we introduce

computers?" but "How should we introduce them?" ([11] as cited in [7]). If coding is conceived as a skill that must begin to be taught early in life [8], and new curricula worldwide in preschool and primary education is covering computational thinking, digital technologies and related areas are being introduced, many preservice teachers are having to undergo professional development to be able to deliver the new material [13]. There are a number of obstacles to bringing coding into the classroom. Even putting obstacles such as the cost to training, teachers have a tendency to teach the way they were taught, and systemwide reform is difficult to implement. To properly bring hands-on learning (or coding, robotics) into the classroom, the classroom must change from a teacher lecturing to a teacher being a mentor [7].

In this paper we described a course that we have developed at the Department of Preschool Education at the University of Crete in an attempt to help preservice teachers to learn CT concepts and programming. Owing to the fact that preservice teachers find it difficult to master the syntax of programming languages in general [23], we believe that the choice of visual programming language is an important factor in learning programming [18]. In this course we chose Scratch as the main programming environment to create an area for preservice teachers for their innovative ideas and a platform to cultivate preservice teachers' computational thinking.

The results, like other studies, show that by enhancing the course curriculum with Scratch and development projects in the Scratch environment, students' performance on CT improved significantly. Similar to Kim et al. [23] research results, we also agree that "Scratch helped pre-service teachers focus on what they could do with programming languages (p. 971)." Scratch helped preservice teachers to overcome their programming difficulties (e.g., syntax) and to focus on core aspects of computational thinking [23].

As it is widely known, changes in learning and teaching practice in class can precede changes in teachers' attitudes and beliefs. Thus, the changes in attitude noted in this study suggest that the preservice teachers believe that Scratch would be a useful tool to do their job and using Scratch would enable them to use technology more effectively [4]. Similar to the study of Arpaci [2], preservice teachers think that using Scratch would increase their productivity, enhance their effectiveness, improve their job performance, and ease their job. Another important thing to consider is that students with no prior programming experience noted that Scratch had assisted them in learning programming.

Based on the success of course, we made the following conclusions:

- We believe that training preservice kindergarten teachers to coding is the best strategy to ensure that all in-service kindergarten teachers will have a technological literacy and computational thinking skills. By introducing coding in university, students will have enough time and exposure to acquire solid computational thinking before they teach in kindergarten.

- The majority of preservice teachers are willing to invest time and effort in training related to CT skills. They recognize that they need to have a technological literacy and computational thinking skills to be prepared for the future.

- There are CT education resources such as lessons and teaching materials available online which are suitable for the novice programmers. Those lessons and teaching resources can be implemented as curriculum which reflects a scaffolding approach.

In future work, it would be an idea to plan out a more open-ended set of challenges, which would allow students to use most advanced CT concepts. Also, it would be a good idea to integrate in the course smart robots such as Bee-Bot and Kibo or Internet-connected smart toys such as Sphero. Also, as a new version of Scratch, Scratch 3.0 is on the way and it would be a good idea to integrate in a new course the use of smart mobile devices such as tablets as a part of new students' experience.

Acknowledgements

We like to thank all of the students involved in this project.

Conflict of interest

There is no conflict of interest to be declared.

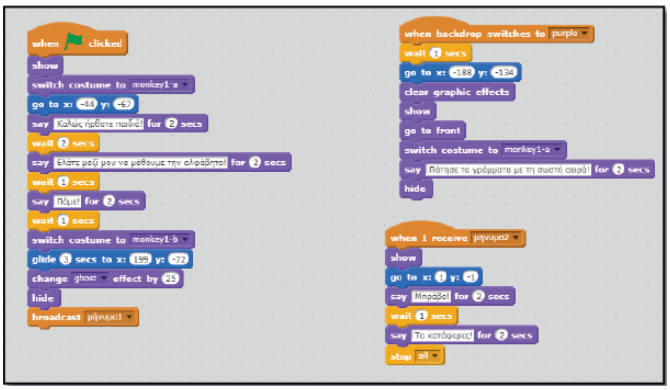
A. Appendix

Example of the user interface, code, and Dr. Scratch scores in two randomly selected students' projects.

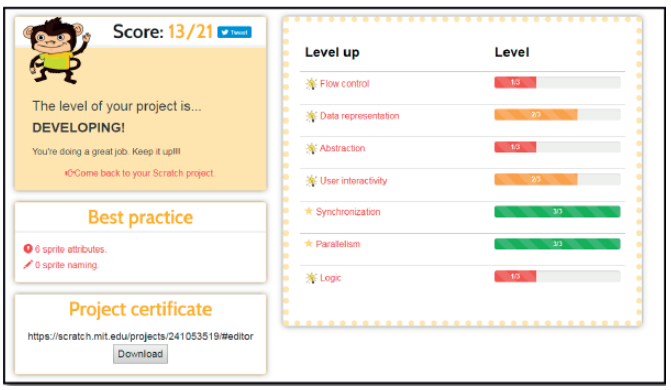
A.1 Project 1. Learn Greek alphabet

A. User interface

B. Code example

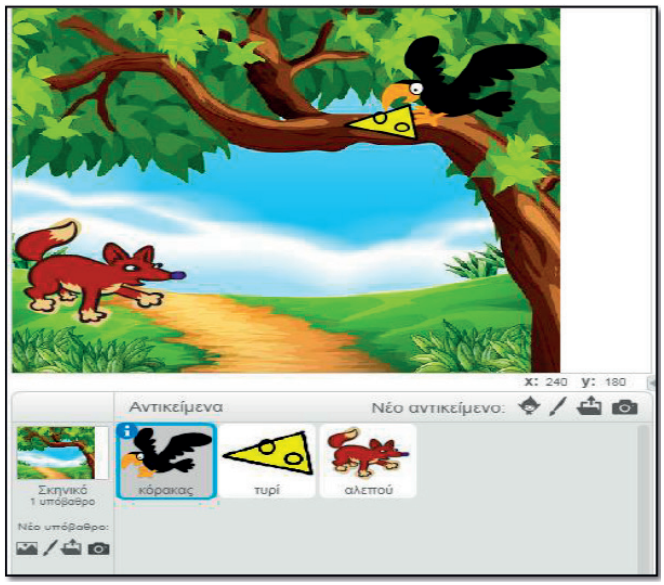


C. Dr. Scratch score

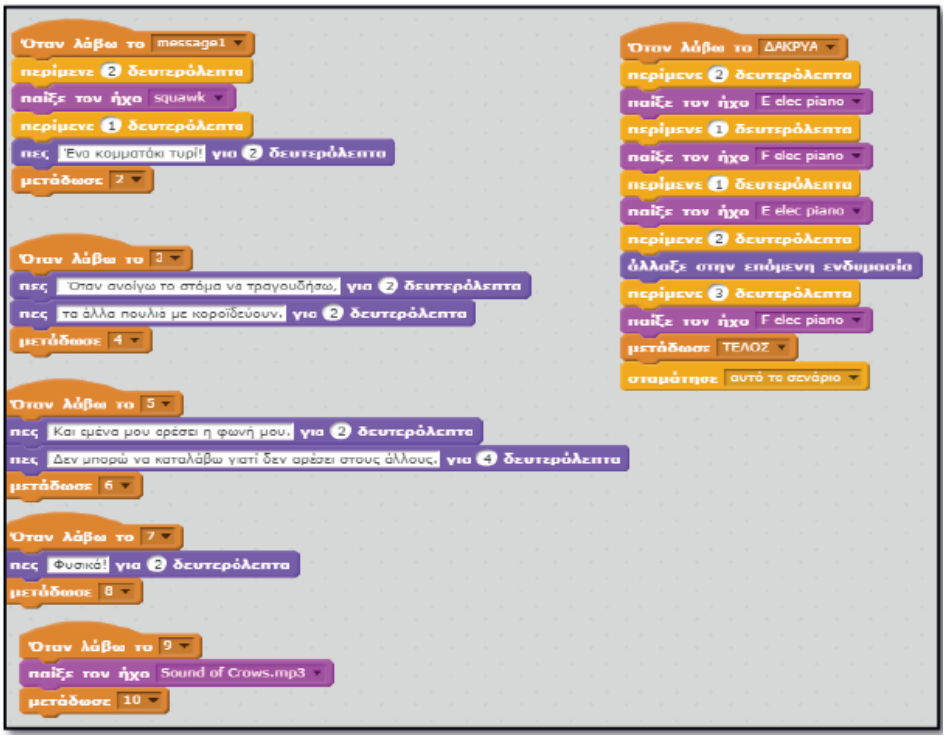


A.2 Project 2. The Fox and the Crow: An Aesop's Fable

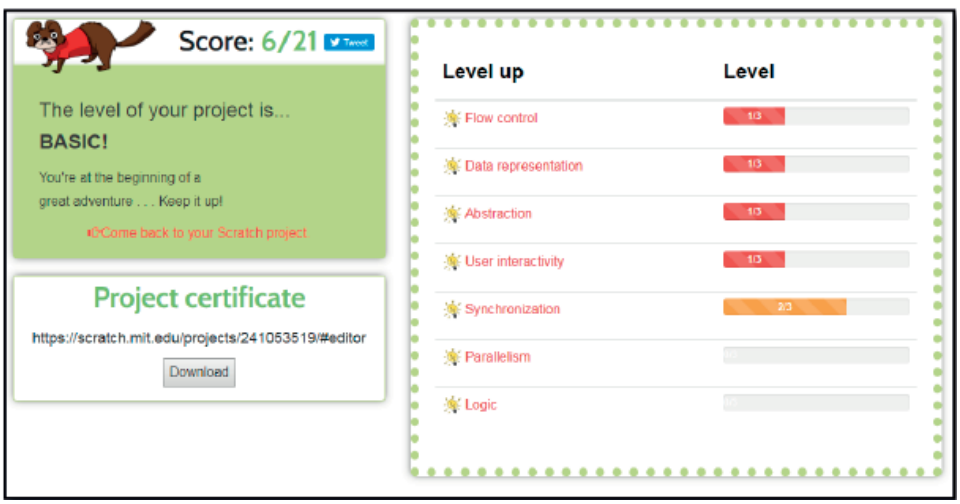
A. User interface



B. Code example



C. Dr. Scratch score



Author details

Stamatios Papadakis and Michail Kalogiannakis*
Department of Preschool Education, Faculty of Education, University of Crete, Crete, Greece

*Address all correspondence to: mkalogian@edc.uoc.gr

References

[1] Agatolio F, Moro M. A workshop to promote Arduino-based robots as wide spectrum learning support tools. In: Merdan M, Lepuschitz W, Koppensteiner G, Balogh R, editors. Robotics in Education. Advances in Intelligent Systems and Computing. Vol. 457. Cham: Springer; 2017. pp. 113-125

[2] Arpaci I. A comparative study of the effects of cultural differences on the adoption of mobile learning. British Journal of Educational Technology. 2015;**46**(4):699-712

[3] Bates D. Raspberry Pi Projects for Kids. Birmingham B3 2PB, UK: Packt Publishing Ltd; 2015

[4] Bean N, Weese J, Feldhausen R, Bell RS. Starting from Scratch: Developing a pre-service teacher training program in computational thinking. In: 2015 IEEE Frontiers in Education Conference (FIE), Camino Real El Paso, El Paso, TX, USA, 2015. pp. 1-8. DOI:10.1109/FIE.2015.7344237

[5] Becker SA, Brown M, Dahlstrom E, Davis A, DePaul K, Diaz V, et al. The NMC Horizon Report: 2018 Higher Education Edition. Austin, Texas: The New Media Consortium: 2018. pp. 1-54

[6] Benton L, Hoyles C, Kalas I, Noss R. Bridging primary programming and mathematics: Some findings of design research in England. Digital Experiences in Mathematics Education. 2017;**3**(2):115-138

[7] Bers MU. Blocks to Robots: Learning with Technology in the Early Childhood Classroom. New York: Teachers College Press; 2007

[8] Bers MU. Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. In: Global Engineering Education Conference (EDUCON). IEEE; 2018. pp. 2094-2102

[9] Bocconi S, Chioccariello A, Dettori G, Ferrari A, Engelhardt K. Developing computational thinking in compulsory education—Implications for policy and practice. Technical report, European Union Scientific and Technical Research Reports, EUR 28295 EN. 2016

[10] Buitrago Flórez F, Casallas R, Hernández M, Reyes A, Restrepo S, Danies G. Changing a generation's way of thinking: Teaching computational thinking through programming. Review of Educational Research. 2017;**87**(4):834-860

[11] Clements DH, Sarama J. Learning Math, Science and Technology is good for preschoolers. 2017. Available from: https://www.childandfamilyblog.com/early-childhood-development/learning-math-science-technology-preschoolershoolers/ [Accessed: 25-06-2018]

[12] Curran J. A guide to programming languages for coding in class. 2017. Available from: https://www.teachermagazine.com.au/articles/a-guide-to-programming-languages-for-coding-in-class [Accessed: 26-06-2018]

[13] Duncan C, Bell T, Atlas J. What do the teachers think?: Introducing computational thinking in the primary school curriculum. In: Proceedings of the Nineteenth Australasian Computing Education Conference. ACM; 2017. pp. 65-74

[14] European Schoolnet. Guidelines for School Leaders. Brussels, Belgium. 2018. http://www.eun.org/documents/411753/1866395/EUN+Annual+Report+2017_Public+Version_FINAL2.pdf/4dffb8dc-cf21-4506-acc1-cf17696a710c

[15] Evangelopoulou O, Xinogalos S. MYTH TROUBLES: An open-source educational game in Scratch for greek mythology. Simulation & Gaming. 2017;**49**(1):71-91

[16] Fesakis G, Serafeim K. Influence of the familiarization with Scratch on future teachers' opinions and attitudes about programming and ICT in education. ACM SIGCSE Bulletin. 2009;**41**(3):258-262

[17] Gomes TCS, Falcão TP, Tedesco PCDAR. Exploring an approach based on digital games for teaching programming concepts to young children. International Journal of Child-Computer Interaction. 2018;**16**:77-84

[18] Iskrenovic-Momcilovic O. Choice of visual programming language for learning programming. International Journal of Computers. 2017;**2**:250-254

[19] Jaipal-Jamani K, Angeli C. Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. Journal of Science Education and Technology. 2017;**26**(2):175-192

[20] Kalelioglu F, Gülbahar Y. The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. Informatics in Education. 2014;**13**(1):33-50

[21] Kalogiannakis M, Papadakis S. Pre-service kindergarten teachers acceptance of "ScratchJr" as a tool for learning and teaching computational thinking and Science education. In: Proceedings of the 12th Conference of the European Science Education Research Association (ESERA), «Research, Practice and Collaboration in Science Education»; 21-25 August 2017. Dublin, Ireland: Dublin City University and the University of Limerick; 2017

[22] Kalogiannakis M, Papadakis S. A proposal for teaching ScratchJr programming environment in preservice kindergarten teachers. In: Proceedings of the 12th Conference of the European Science Education Research Association (ESERA), «Research, Practice and Collaboration in Science Education»; 21-25 August 2017. Dublin, Ireland: Dublin City University and the University of Limerick; 2017:965-982

[23] Kim H, Choi H, Han J, So HJ. Enhancing teachers' ICT capacity for the 21st century learning environment: Three cases of teacher education in Korea. Australasian Journal of Educational Technology. 2012;**28**(6)

[24] Mackintosh J, White E, Dickerson C. Developing teachers as leaders of science in primary schools. Journal of Emergent Science. 2017;**12**:64-71

[25] Martínez-Valdés JA, Velázquez-Iturbide JÁ, Hijón-Neira R. A (relatively) unsatisfactory experience of use of Scratch in CS1. In: Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17); 18-20 October 2017; Cádiz, Spain (Article. 8). New York, NY, USA: ACM; 2017

[26] Moreno-León J, Robles G. Code to learn with Scratch? A systematic literature review. In: Global Engineering Education Conference (EDUCON), 2016 IEEE. IEEE; 2016. pp. 150-156

[27] Moreno-León J, Robles G, Román-González M. Towards data-driven learning paths to develop computational thinking with Scratch. IEEE Transactions on Emerging Topics in Computing. 2017. DOI:10.1109/TETC.2017.2734818

[28] Papadakis S. Creativity and innovation in European education. 10 years eTwinning. Past, present and the future. International Journal of Technology Enhanced Learning. 2016;**8**(3-4):279-296

[29] Papadakis S. The use of computer games in classroom environment. International Journal of Teaching and Case Studies. 2018;**9**(1):1-25

[30] Papadakis S. Is pair programming more effective than solo programming for secondary education novice programmers?: A case study. International Journal of Web-Based Learning and Teaching Technologies. 2018;**13**(1):1-16

[31] Papadakis S, Kalogiannakis M. Using gamification for supporting an introductory programming course. The case of ClassCraft in a secondary education classroom. In: Proceedings of the 2nd EAI International Conference on Design, Learning and Innovation; 30-31 October 2017; Heraklion, Greece. 2017

[32] Papadakis S, Orfanakis V. The combined use of lego mindstorms NXT and app inventor for teaching novice programmers. In: Alimisis D, Moro M, Menegatti E, editors. Educational Robotics in the Makers Era. Edurobotics 2016. Advances in Intelligent Systems and Computing. Vol. 560. Cham: Springer; 2016. pp. 193-204

[33] Papadakis S, Orfanakis V. Comparing novice programing environments for use in secondary education: App Inventor for Android vs. Alice. International Journal of Technology Enhanced Learning. 2018;**10**(1-2):44-72

[34] Papadakis S, Kalogiannakis M, Zaranis N. Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: A case study. International Journal of Mobile Learning and Organisation. 2016;**10**(3):187-202

[35] Papadakis S, Kalogiannakis M, Orfanakis V, Zaranis N. The appropriateness of Scratch and app inventor as educational environments for teaching introductory programming in primary and secondary education. International Journal of Web-Based Learning and Teaching Technologies. 2017;**12**(4):58-77

[36] Papadakis S, Kalogiannakis M, Orfanakis V, Zaranis N. Novice programming environments. Scratch and app inventor: A first comparison. In: Fardoun HM, Gallud JA, editors. Proceedings of the 2014 Workshop on Interaction Design in Educational Environments. New York: ACM; 2014. pp. 1-7

[37] Papavlasopoulou S, Sharma K, Giannakos MN. How do you feel about learning to code? Investigating the effect of children's attitudes towards coding using eye-tracking. International Journal of Child-Computer Interaction. 2018:**17**:50-60

[38] Redecker C. European Framework for the Digital Competence of Educators: Digital Competence Framework for Educators . Punie Y. (ed). EUR 28775 EN. Publications Office of the European Union, Luxembourg, 2017. ISBN 978-92-79-73494-6, DOI:10.2760/159770, JRC107466

[39] Resnick M, Maloney J, Monroy-Hernández A, Rusk N, Eastmond E, Brennan K, et al. Scratch: Programming for all. Communications of the ACM. 2009;**52**(11):60-67

[40] Scratch-wiki. Scratch 3.0. 2018. Available from: https://en.scratch-wiki.info/wiki/Scratch_3.0 [Accessed: 25-06-2018]

[41] Smith S, Burrow LE. Programming multimedia stories in Scratch to integrate computational thinking and writing with elementary students. Journal of Mathematics Education. 2016;**9**(2):119-131

[42] Strawhacker A, Lee M, Bers M. Teaching tools, teachers' rules: Exploring the impact of teaching styles

on young children's programming knowledge in ScratchJr. International Journal of Technology and Design Education. 2018;**28**(2):347-376

[43] Topalli D, Cagiltay NE. Improving programming skills in engineering education through problem-based game projects with Scratch. Computers & Education. 2018;**120**:64-74

[44] Tsur M. Scratch Microworlds: Introducing novices to Scratch using an interest-based, open-ended, scaffolded experience [doctoral dissertation]. Massachusetts Institute of Technology; 2017

[45] Weintrop D, Hansen AK, Harlow DB, Franklin D. Starting from Scratch: Outcomes of early computer science learning experiences and implications for what comes next. In: Proceedings of the 2018 ACM Conference on International Computing Education Research. ACM; 2018. pp. 142-150

[46] Wilson A, Moffat DC. Evaluating Scratch to introduce younger school children to programming. In: Proceedings of the 22nd Annual Psychology of Programming Interest Group. Leganés, Spain: Universidad Carlos III de Madrid; 2010. p. 14

[47] Wing JM. Computational thinking. Communications of the ACM. 2006;**49**(3):33-35