Education and Information Technologies
https://doi.org/10.1007/s10639-018-9823-2

A blended learning course for playfully teaching programming concepts to school teachers

Fotis Lazarinis[1] · Christoforos V. Karachristos[1] · Elias C. Stavropoulos[1] · Vassilios S. Verykios[1]

Received: 19 June 2018 / Accepted: 28 September 2018/Published online: 11 October 2018

Abstract
In this paper we report our experiences from a University outreach program with primary and secondary education teachers of various specialties. Our goal was to improve the coding abilities of teachers through Scratch activities. The participants can in turn teach their students, multiplying that way the benefitted population. To increase the participation and the completion percentage, the activities are designed as a course in Moodle realized in discrete runs with manageable groups, in a blended learning approach. The educational material was a combination of learning objects with specific objectives, video material and try-out activities. The course has been completed by 559 teachers from various Greek districts, mainly of regional areas, with a high completion rate of 65%. The participants found the experience highly satisfying, interesting and agreed that they had been supported effectively throughout the process. In the paper we describe the rational of our approach, the design and implementation phases of the project, the outcomes and the main findings of the evaluation of the user opinions.

Keywords Computer science education · Computational thinking · Multimedia learning · Professional teacher development · Scratch · Blended learning · MOOC

✉ Fotis Lazarinis
fotis.lazarinis@ac.eap.gr

Christoforos V. Karachristos
karachrist@eap.gr

Elias C. Stavropoulos
estavrop@eap.gr

Vassilios S. Verykios
verykios@eap.gr

[1] Hellenic Open University, Patras, Greece

1 Introduction

It is well established that computational thinking is a basic skill that should become an integral part of every child's education (Wing 2006). Although there is some vagueness around the term "computational thinking", an acceptable definition could be "Computational thinking as the application of high level of abstraction and an algorithmic approach to solve any kind of problems" (García-Peñalvo 2016). In other words, any activity that promotes the application of logical, structured and modular solutions to various problems could fall under this broad term. Coding activities is the primary route to improve the algorithmic thinking of individuals (Lye and Koh 2014).

Several initiatives have been launched to reinforce the coding skills of students and teachers using a mixture of pedagogical approaches and enabling technologies. Universities, schools, and other associations participate in local actions, funded programs and international schemes like the Code week and the Hour of code. A project exposing computational thinking to teachers and students has been implemented in Switzerland (Lamprou et al. 2017). Through intensive face-to-face sessions and online and paper based material the intended concepts have been taught. A problem-based approach is considered as inherent in CS teaching (Armoni 2017). A project using the Alice software for teaching computer science concepts to middle school students has been realized in Qatar (Al Sabbagh et al. 2017). Teaching material has been created to aid the participating teachers to deliver the course. The authors argue that a successful reform initiative must begin with recognizing the importance of teachers in raising student performance. Therefore they trained teachers to subsequently teach students. A CS curriculum for "middle-years students," grades 4–10 in US, is developed in the MyCS project (Castro et al. 2016). The work presents the experiences of the five-year development and deployment of the project. MyCS consists of different modules emphasizing on computational thinking and problem solving. Scratch is used as the main programming environment. Teaching and learning of computer programming at the university level has been developed within the Second Life virtual world (Esteves et al. 2010). Their results suggest that it is possible to employ virtual environments for better effectiveness in teaching and learning of programming.

Visual "block" programming tools like Scratch, Alice, and App Inventor have been extensively employed for introducing programming to students and teachers, e.g., (Castro et al. 2016). In a recent survey, students agreed that blocks-based programming is easy due to the natural language description of the blocks, the drag-and drop interaction, and the ease of browsing to the available commands (Weintrop and Wilensky 2015). School teachers have been introduced to programming through visual environments, e.g. (Al Sabbagh et al. 2017). In (Rodger et al. 2009) middle school students have been involved in one-week summer camps of instruction in Alice. The teachers created lesson plans with Alice worlds to interactively teach a topic. The participants used a large variety of basic Alice concepts and computer science concepts in the worlds they built. Teachers have enthusiastically participated in the workshops. The project involved students with similar positive opinions.

Scratch has been also used in many projects realized from Universities for secondary education students and teachers and even for tertiary education students.

In (Dee et al. 2017) a project concerning the development of re-usable activities in Scratch is presented. These activities can be run in schools by teachers, or run by university staff in schools as outreach activities. The experiences of using Scratch in classroom practice, for two years, in five different schools in Spain are discussed in (Sáez-López et al. 2016). Students considered Scratch activities to be motivational and enjoyable. The authors observed an advancement in their computational thinking. Their intervention took place in two academic years in which they analyzed the practice of integrating coding and visual blocks programming in sciences and arts.

Our focus in this work is to help primary and secondary education teachers to acquire a solid understanding of the basic programming concepts which are common in all programming languages. In the rest of the paper we present the design principles of the project, the pedagogical approach, the implementation details and the results of a user opinion survey.

2 Who, how, and what to teach?

2.1 Target group

Our main aim was to improve the computational thinking in primary and middle schools through university outreach activities regarding coding activities. Although there are elective and compulsory ICT courses in all primary and secondary education classes in Greece, the curricula focus mainly on the usage of computer applications and on theoretical concepts. In General Secondary Education only a small percentage of school students occupy themselves with algorithmic activities at the last year of the upper high school. Students who take University admission exams targeting at specific technical University departments have to encode their solutions in pseudocode, composed though in paper and not in computers. Students attending Technical Secondary Institutions take more programming courses. However, the student population attending technical schools in Greece, in comparison with those attending the general secondary education, is small. All in all, only a small percentage of the students in primary and secondary education are exposed to computational thinking through coding activities.

Running a university outreach program with the purpose of helping students to comprehend basic computer concepts could involve only a small number of students. The admission of external staff to a Greek school for running any kind of instructional activity has to undergo a review process at a central level from the educational authorities. Running the activities in distance learning mode would make the process difficult, as the children are not accustomed to this kind of learning process which requires self- motivation and dedication.

To overcome these difficulties and still affect a large number of individuals we decided to engage mainly teachers. The basic idea behind our work was to train a significant number of school teachers who will in turn transfer their skills to their students, multiplying the benefited population. Educating the teachers using asynchronous online material and practical assignment in a MOOC-ish way would increase the number of participants.

2.2 Another MOOC?

The literature has shown that MOOCs have a low completion rate (Liyanagunawardena et al. 2013). In a large scale survey with data from 91 MOOCs it was shown that the completion rates ranged from 0.9 to 36.1%, with a median value of 6.5% (Jordan 2014). One of the dropout reasons is the lack of student support (Onah et al. 2014). The low teacher to student ratio is another reason which leads students to leave a MOOC (Guo and Reinecke 2014). Examples of problems like "student posts to the forums are often ignored or not addressed in a timely manner" have been identified in relevant studies (Zheng et al. 2015). The lack of interacting with other students and teachers is another factor affecting the successful completion of MOOCs (Eriksson et al. 2017).

Therefore, developing another typical MOOC would not adhere to our goal to educate a large number of teachers, as the lack of support and interaction would lead to a low completion rate. It was thus decided to design and implement an e-course with increased support towards the participants in order to improve the completion rate and support the needs of the learners. A blended learning approach, where the instructional process comprises face-to-face meetings, online asynchronous video lectures, synchronous sessions and focused support to the participants via forums and email, is thus preferable. This approach would allow us to teach an acceptable number of teachers, mainly in a distance learning mode, and still have the overall control of the process.

2.3 Topics and programming environment

The next phase of the project was to decide the topics to be taught. As mentioned before we wanted to help the learners understand the process of programming and the basic concepts present in all programming languages. The core concepts are *initialization*, *sequence*, *selection*, *iteration*, and the *usage of variables* which are the common functions are present in every programming environment. By solving computer problems of gradually increasing difficulty the participants develop higher order skills like abstraction, decomposition, and recursion. These are the core elements in computational thinking and, in general, they represent the modern skills needed for solving real world problems. So we decided to develop material consisting of well-structured activities aiming at promoting these skills. Each activity should target at one or more of these skills and be composed of steps so that the participants can understand the notion of decomposition. Scaffolding activities help them to develop their skills further by gradually reducing the level of support. For example, at first a problem is presented and explained to the learners in a step-by-step mode. Then they are asked to solve a slightly different problem using the acquired knowledge or to make some amendments to the original solution in order to better understand the role of each command in a sequence of commands.

The literature review has shown that visual "block" programming tools are the most suitable approach to introduce programming to non-technical users. Among these tools, we consider Scratch to be the most appropriate for our aims, as it encompasses all the necessary elements and techniques; it is quite easy to learn; one can start building a meaningful small application even from the first class and can therefore remain motivated and enthusiastic; there are online and offline versions which make it easier to use the environment for different educational purposes; and finally it has been used in most educational activities in Greece for introducing programming.

3 Code-create-play with scratch: The 'SCRATCHCODING' project

3.1 Requirements

Having decided on the main issues of the project, we needed an efficient design that would include all the conclusions described in the previous sections. Being experienced in distance learning education as the primary distance learning providing University in Greece and having performed an analysis phase, through informal interviews with representative students, we concluded that the system should meet the following core functional requirements:

- The teaching material shall have a game-ish look and feel
- The course shall be modular
- Each learning module shall:

– have specific aims and prerequisites
– be presented in form of video lectures
– include textual versions of the teaching contents
– include try-out activities
– include quizzes which require some programming task(s) in order to be answered
– include programming exercises
– include external resources

- A face-to-face meeting shall occur at the beginning of the course
- Users shall be supported by email, forum and peers
- At least one synchronous sessions shall take place during the course to address questions and problems of the questions
- Additional synchronous sessions should occur during should the participants ask for it
- The learning platform shall be widespread, so that (most) users would be familiar with it
- Users shall register to the system and their data shall be verified
- Formal certification shall be available to those who complete all the tasks
- The course shall be run in cycles with small populations for the better support of the users

The non-functional requirements for our system are generic and common in every information system. The system must be reliable, user friendly, fast, available, and able to handle errors effectively.

3.2 Learning material & instructional methodology

Based on the requirements identified in the previous section, we designed a modular e-course called "Scratchcoding: Code-Create-Play with Scratch". Figure 1 depicts its organizational structure. The educational material consists of learning objects organized into larger units of related topics which in turn form the e-course. Each learning object contains specific objectives, required knowledge, teaching material in video and textual forms, and activities, quizzes, and exercises which require of users to run, and to complete coding blocks or to develop full applications. Links to other resources are also

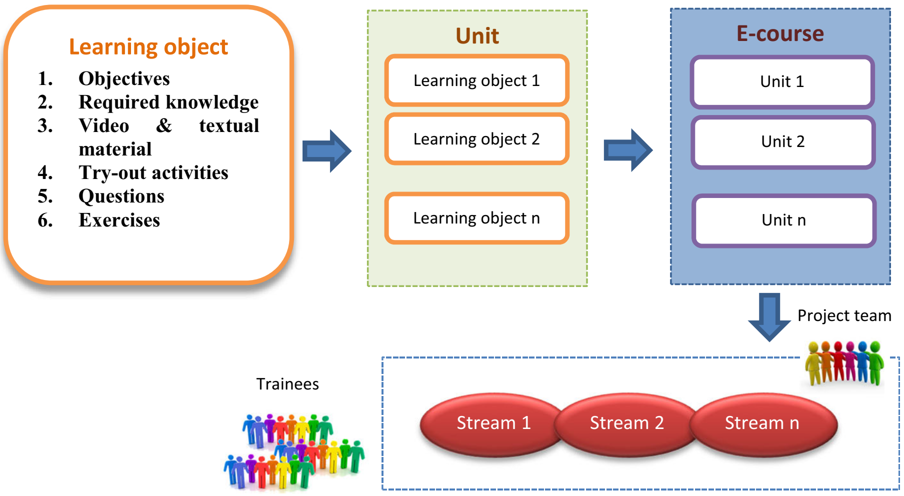


Fig. 1 Organization of the course in units run in different streams

part of a learning object. The modular architecture of the educational content increases its reusability and shareability.

The instructional methods applied within a learning object are 'learning by example' and 'problem solving'. The videos discuss and explain specific examples with step-by-step instructions and provide analytical explanations of the functions of the visual blocks, the characters and the rest of the structural elements of a Scratch story. Students acquire the expected skills by repeating the examples which can be run in the online or the offline Scratch versions. The video lectures act as facilitators promoting self-learning, helping students to develop critical thinking skills. Various questions are posed and answered in the videos directing students to the way we think when we develop computer programs or when we algorithmically deal with problems. The try-out exercises, the quizzes and the assessment activities increase the active participation of the learners and follow a 'problem solving' approach. Easy or more difficult problems are posed, which require specific and measurable actions on behalf of the learners. These scaffolding activities improve gradually the coding skills of the learners and help them promote their computational thinking. To help the participants further, some of the activities have a solution attached which could be used in case someone could not make a progress on their own.

Along with the learning environment, users have to eventually use the online version of Scratch. For the quizzes and assignments that must be completed, learners have to create or complete a Scratch program in scratch.mit.edu in specific collections and submit their link to the learning platform. That way we can check their actions, assess their code and provide useful feedback, forcing them to be actively involved into coding. Students can perform some of the activities in the offline version of Scratch, but they have to eventually upload them to the online version and share it with their peers and the rest of the Scratch community.

The course is designed to run in different streams with a goal of 150–200 participants in each stream. Having a manageable number of learners helps us to provide focused feedback in a timely manner and handle any problems that might arise. Each stream starts with a face-to-face meeting with the participants. In this session, the aims of the outreach program are explained; we demonstrate the learning platform, the Scratch online environment and the working method; we explain the teaching methodology; we explain the requirements and we address any technical or other issues raised by the participants. These direct instruction meetings host 25–30 teachers so as to be useful and therefore 4–5 meetings per course cycle occur. One synchronous meeting with optional attendance takes place during the course, so as to discuss any issues and to motivate the students. Communication through email and forum is another way which supports users. Peer assessment is also available for most of the activities.

3.3 Implementing the e-course

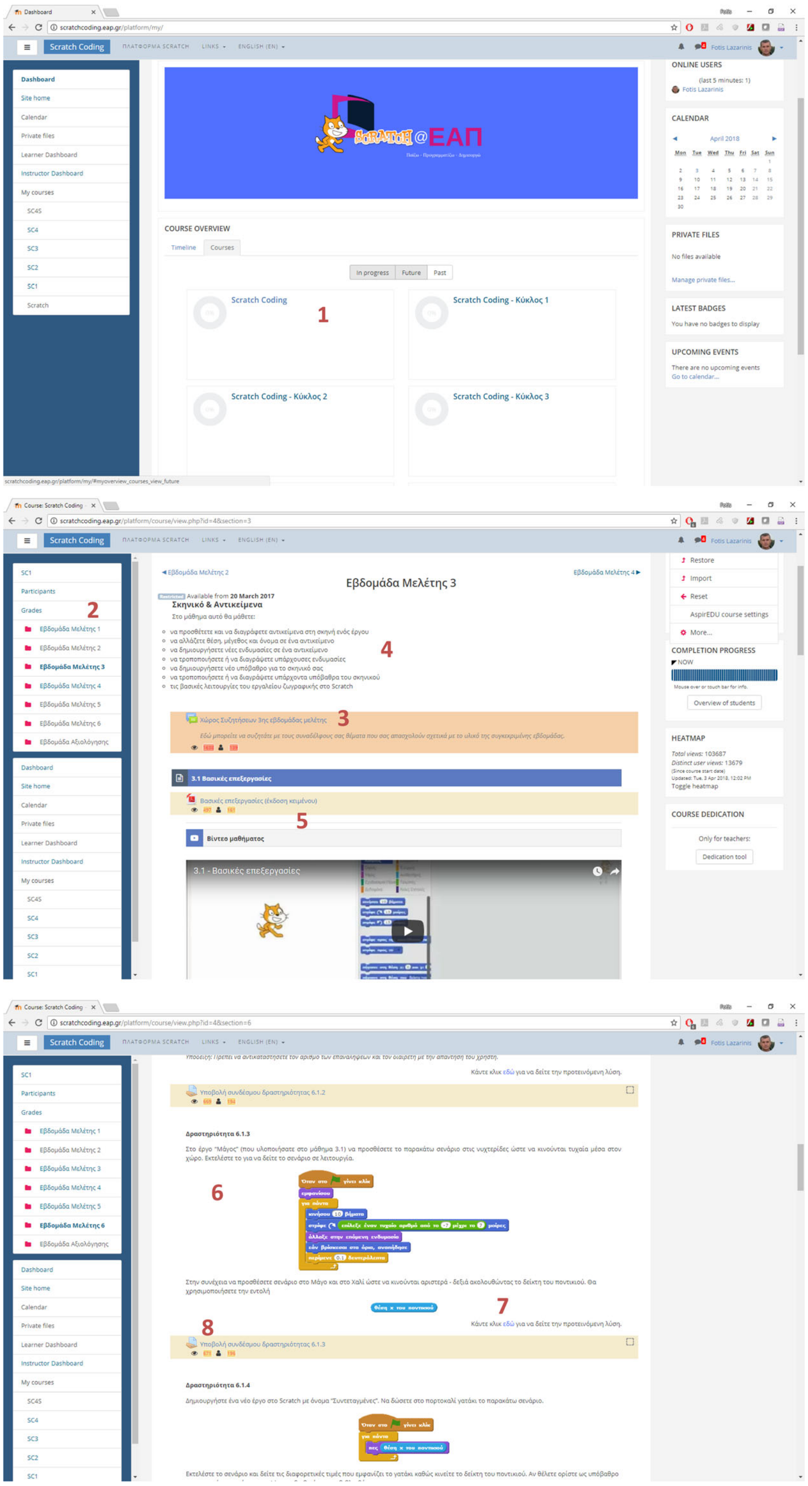
The next stage was the implementation of the course. The educational material is integrated into a Moodle course following a "topics format", as each unit, based on its difficulty may take different amounts of time to complete. Each Moodle topic corresponds to one unit of the design shown in Fig. 1. Moodle is the most common learning platform, hence many teachers are familiar with it and therefore its selection satisfies the respective functional requirement. The abilities of the tool to present the material in multimedia form (rich text, html, video, interactive activities, etc.), to monitor the progress of the students and to support the participants through fora and synchronous communication support effectively our technical demands.

Figure 2 shows three screenshots of the course in Moodle (http://scratchcoding.eap.gr). In point #1 we can see that the course is replicated to run in cycles. #2 shows the contents of the course, i.e. the units wrapping the related learning objects. In #3 we see the forum and in #4 the objectives of the specific unit. In area #5 the textual and the video material of a learning object exist. In #6 we can see a "try-out" activity with a solution linked in #7. The link of the activity in scratch.mit.edu has to be submitted in area #8. The applications are shared in collections like https://scratch.mit.edu/studios/3510745/ and https://scratch.mit.edu/studios/3510814/.

Objects in points #4 to #7 along with some additional quizzes and activities which are not shown in Fig. 2, belong to one learning object which is numbered as 3.1 in the e-course, meaning that it is the learning object 1 of unit 3. This learning object could be re-used in other e-learning applications. More learning objects are included in the same unit. Areas #1, #2 and #8 are not part of the learning object, but they belong to the learning platform for supporting the teaching process. Prior to launching the course, the e-learning application has been tested with representative users to assess its suitability and identify potential problems.

3.4 Running the blended course in streams

The material and the e-course has been developed between November 2016 and March 2017 and the outreach activities have been deployed in 4 different streams involving various regional areas of Greece from March 2017 to December 2017, with a break during the school summer holidays. Each stream lasted 2 months. The capital and a few other major cities have been excluded, to train teachers from remote and rural areas which have less training opportunities. The project has been advertised through administrative channels pertaining secondary education institutes and through



◄ Εβδομάδα Μελέτης 2 Εβδομάδα Μελέτης 4 ►

Εβδομάδα Μελέτης 3

Restricted Available from **20 March 2017**

Σκηνικό & Αντικείμενα

Στο μάθημα αυτό θα μάθετε:

- να προσθέτετε και να διαγράφετε αντικείμενα στη σκηνή ενός έργου
- να αλλάζετε θέση, μέγεθος και όνομα σε ένα αντικείμενο
- να δημιουργήσετε νέες ενδυμασίες σε ένα αντικείμενο
- να τροποποιήσετε ή να διαγράψετε υπάρχουσες ενδυμασίες
- να δημιουργήσετε νέο υπόβαθρο για το σκηνικό σας
- να τροποποιήσετε ή να διαγράψετε υπάρχοντα υπόβαθρα του σκηνικού
- τις βασικές λειτουργίες του εργαλείου ζωγραφικής στο Scratch

Χώρος Συζητήσεων 3ης εβδομάδας μελέτης

Εδώ μπορείτε να συζητάτε με τους συναδέλφους σας θέματα που σας απασχολούν σχετικά με το υλικό της συγκεκριμένης εβδομάδας.

3.1 Βασικές επεξεργασίες

Βασικές επεξεργασίες (έκδοση κειμένου)

Βίντεο μαθήματος

3.1 - Βασικές επεξεργασίες

Υπόδειξη: Πρέπει να αντικαταστήσετε τον αριθμό των επαναλήψεων και τον διαιρέτη με την απάντηση του χρήστη.

Κάντε κλικ εδώ για να δείτε την προτεινόμενη λύση.

Υποβολή συνδέσμου δραστηριότητας 6.1.2

Δραστηριότητα 6.1.3

Στο έργο "Μάγος" (που υλοποιήσατε στο μάθημα 3.1) να προσθέσετε το παρακάτω σενάριο στις νυχτερίδες ώστε να κινούνται τυχαία μέσα στον χώρο. Εκτελέστε το για να δείτε το σενάριο σε λειτουργία.

```
Όταν στο 🏁 γίνει κλικ
εμφανίσου
για πάντα
    κινήσου 10 βήματα
    στρίψε ↻ επίλεξε έναν τυχαίο αριθμό από το -7 μέχρι το 7 μοίρες
    άλλαξε στην επόμενη ενδυμασία
    εάν βρίσκεσαι στα όρια, αναπήδησε
    περίμενε 0.1 δευτερόλεπτα
```

Στην συνέχεια να προσθέσετε σενάριο στο Μάγο και στο Χαλί ώστε να κινούνται αριστερά - δεξιά ακολουθώντας το δείκτη του ποντικιού. Θα χρησιμοποιήσετε την εντολή

`θέση x του ποντικιού`

Κάντε κλικ εδώ για να δείτε την προτεινόμενη λύση.

Υποβολή συνδέσμου δραστηριότητας 6.1.3

Δραστηριότητα 6.1.4

Δημιουργήστε ένα νέο έργο στο Scratch με όνομα "Συντεταγμένες". Να δώσετε στο πορτοκαλί γατάκι το παρακάτω σενάριο.

```
Όταν στο 🏁 γίνει κλικ
για πάντα
    πες θέση x του ποντικιού
```

Εκτελέστε το σενάριο και δείτε τις διαφορετικές τιμές που εμφανίζει το γατάκι καθώς κινείτε το δείκτη του ποντικιού. Αν θέλετε ορίστε ως υπόβαθρο

Fig. 2 Screenshots of the "scratchcoding" course in Moodle

educational portals. In total among 2357 applications we selected 860 teachers, which was a number that we could manage in the first phase of the project, taking into account that a percentage of the selected teachers would not start the course eventually. The teachers were selected on a first-applied-first-selected basis, in such a way though, so that a balance could be kept between genders, specialties and the location of the school. Thus we maintained an equal distribution between women and men and of various specialties (e.g. primary education teachers, mathematicians, language teachers, etc.). The participants needed basic ICT skills to complete the course. Therefore there was no other demands related to previous knowledge or advanced technical skills and so no full profiles have been acquired for each participant for the process to be simplified.

Of these 860 persons who have been enrolled in the course, 243 have never started the course and finally 559 actually completed the course and received their free certification of achievement. Estimating the success based on the registered teachers, we had a completion rate of 65% (559/860). However it has been noted that the definition of completion rate as a percentage of enrolled students may be over-simplistic (Ho et al. 2015) and therefore alternative methods have been proposed. If we measure the success as a percentage of those who completed at least one activity, then the success rises up to 90,60% (559/617). In any case, even with the first method the completion rate is quite high compared to the rates reported in the literature.

To complete the course successfully, the teachers had to complete all the tasks at the end of each unit. These tasks were either try-out activities or tests with closed-ended questions. Their progress was monitored through the reports of Moodle. The teachers had to achieve a passing score (grade $\geq$ 60%) to continue to the next unit. Furthermore, they had to submit an assignment at the end of the course at scratch.mit.edu, which was manually assessed by a member of our team. Feedback was given to each participant and then they were advised to update their solution if it was judged necessary.

During the realization of the course we received various questions of technical nature related, mainly, to how to complete specific steps in an activity. These questions have been answered via the forum by their peers or by the members of our team. At the beginning of the course we dealt with registration issues which have been solved during the face-to-face inaugural session or through email by our technical staff.

Another important outcome of the project has been the fact that few teachers have already used Scratch in their classes. 271 primary and secondary education students have already been taught running selected applications from our material. Additionally, it has already been reported that the students created 39 new Scratch applications apart from having carried out some of the course's application. We expect this number to increase significantly in the next few months.

4 Evaluation

At the end of the course, the participants had to fill out an anonymous questionnaire in order to complete the course. Most of the questions were in a five-Likert scale "Very poor", "Poor", "Acceptable", "Good", "Very Good". Table 1 groups the questions which relate to the material, the organization and the learning environment. All the

Table 1 Questions about the objectives, the material, the learning environment and the organization

Question	A	B	C
The objectives of the course were clear	0.54%	0.89%	98.57%
The learning material was comprehensible	0.00%	1.07%	98.93%
The lessons were well organized and easy to follow	0.00%	4.83%	95.17%
The goals of the course have been achieved	0.00%	2.68%	97.32%
The course was interesting	0.00%	2.86%	97.14%
The learning environment was easy to use	0.00%	3.22%	96.78%
The course had a playful look and feel	0.00%	1.61%	98.39%
There was support from the organizers	0.36%	3.22%	96.42%
The communication with the organizers was immediate	0.54%	3.40%	96.06%

A: Very poor/poor, B: Acceptable, C: Good/Very good

participants had a positive or very positive opinion. They found the course interesting, easy to follow and with substantial support by the organizers. Questions #1 to #5 have been asked per unit as well, with similar results though.

Table 2 shows that most of the participants did not know how to program in Scratch or how to program at all. Nevertheless, the vast majority of the participants agreed that they can now program in Scratch and create new applications. They are willing to use Scratch in their lessons, to suggest the course to another colleague and to participate in a new course with more advanced topics in Scratch. The fact that most of the participants were either novice programmers or did not know how to program at all, strengthens the results in Table 1.

At the end of the survey, the teachers could identify the most positive and the most negative aspect of the course and to suggest changes and additions. Less than half of the participants commented on these free-text questions. A point of concern raised by many participants is associated with the last course unit. Many participants noticed that there was an augmented number of more demanding activities in this unit compared to the previous sections. Therefore they considered that more time was necessary for this unit.

Table 2 Questions about the programming skills

Question	A	B	C
I knew how to program	47.23%	11.09%	41.68%
I knew about Scratch	61.36%	15.56%	23.08%
I knew how to program in Scratch	80.32%	8.77%	10.91%
I learned to program in Scratch	1.61%	10.20%	88.19%
I feel that I can create new Scratch applications	0.00%	15.56%	84.44%
I will use Scratch in my classes	6.98%	14.49%	78.53%
I will suggest the e-course to a colleague	0.00%	4.57%	95.43%
I am interested in a new Scratch course with advanced topics	0.00%	4.06%	95.94%

A: Very poor/poor, B: Acceptable, C: Good/Very good

Almost all of the participants who filled out these free text made quite positive remarks and commented enthusiastically on the organization, the support, and the joyful learning. A few participants asked for more advanced exercises, while some other asked for less exercises. This controversy originates from the previous knowledge of the learners. Those with some programming skills, progressed faster and needed more exercises. Teachers from less technical specialties had more difficulties especially in the last unit. This problem could have been managed with the integration of optional activities, completed only by those who achieved a faster progress.

It is worth mentioning here – although it is not the purpose of this work – that Moodle platform can offer a holistic view of participants' activity and an overall evaluation of the content, by setting up and utilizing learning analytics dashboards (LADs), i.e., certain plug-ins for visualizing the analysis results obtained by the logging records. Using LADs, a course designer or an instructor can have direct feedback related to participants' progress, dedication and engagement, forum participation, access to the educational material and external resources, submission of assignments and quizzes, etc. The reader is referred, for example, to some recent works (Gkontzis et al. 2017a, b) for further reading.

5 Discussion

In this paper we have presented a project aiming at improving the coding skills of primary and secondary education teachers. The activity was implemented as a Moodle course with additional face-to-face support, so as to increase the active participation and improve the completion rate. The learning material has been developed in a modular approach and is well documented with specific objectives and prerequisites, so as to be easily reusable. By following a 'learning by example' and 'problem solving' approaches within a learning object, teachers of various, primarily non-technical, specialties managed to learn the basics of programming in a playful mode. Their evaluation towards our initiative was very positive and the trainees would like to participate in another learning activity with more advanced topics. In total 559 teachers and 271 students have been, so far, affected by our outreach activity.

We plan to repeat the activity in the next school year with some extensions and modifications to the learning material. More optional activities will be added, to maintain the interest of the most accomplished learners. To improve the support towards new learners further, we will engage teachers who have completed a previous run of the course, to help us, on a voluntary basis.

A basic conclusion from our engagement in this program is that the community of teachers are eager for such activities, as professional development of teachers in a changing working environment is important. Understanding the way computers work is a core issue in the digital era. Scratch is a very efficient environment for introducing programming concepts and this was obvious by the comments we received either in the formal survey or through email or from the postings to our social media accounts. Scratch coding activities improves the computational thinking, which was our initial goal, as one has to think in terms of abstraction, decomposition and recursion to develop an application. The visualization of the actions is an easy way to spot execution problems and to work towards an efficient solution.

Acknowledgments This work has been supported by the Stavros Niarchos Foundation.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

Al Sabbagh, S., Gedawy, H., Alshikhabobakr, H., and Razak, S. (2017). Computing curriculum in middle schools: An experience report. *Proceedings of the 22th ACM annual conference on innovation and technology in computer science education (ITiCSE 2017)* 13–16. New York: ACM.

Armoni, M. (2017). Computing in schools: On teaching problem-solving paradigms in K-12 education. *ACM Inroads, 8*(1), 30–32. https://doi.org/10.1145/1118178.1118215.

Castro, B., Diaz, T., Gee, M., Justice, R., Kwan, D., Seshadri, P., and Dodds, S. (2016). MyCS at 5: Assessing a middle-years CS curriculum. In *Proceedings of 47th technical symposium on computer science education (SIGCSE 2016)* 558–563. New York,: ACM.

Dee, H., Cufi, X., Milani, A., Marian, M. and Poggioni V. (2017). Playfully coding: Embedding computer science outreach in schools. In *Proceedings of the 2017 ACM conference on innovation and technology in computer science education (ITiCSE '17)* 176–181. New York,: ACM.

Eriksson, T., Adawi, T., & Stöhr, C. (2017). Time is the bottleneck: A qualitative study exploring why learners drop out of MOOCs. *Journal of Computing in Higher Education, 29*(1), 29–133. https://doi.org/10.1007/s12528-016-9127-8.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2010). Improving teaching and learning of computer programming through the use of the second life virtual world. *British Journal of Educational Technology, 42*(4), 624–637. https://doi.org/10.1111/j.1467-8535.2010.01056.x.

García-Peñalvo, F. J. (2016). What Computational Thinking is. *Journal of Information Technology Research, 9*(3), v–viii.

Gkontzis, A. F., Karachristos, C. V., Lazarinis, F., Stavropoulos, E.C., and Verykios, V.S. (2017a) Assessing Student Performance by Learning Analytics Dashboards. In *Proc. of ICODL2017, SECTION A: theoretical papers, original research and scientific articles* 9(6B):101–115. Greece: Athens.

Gkontzis, A.F., Panagiotakopoulos, C. T., Stavropoulos, E. C. and Verykios, V.S. (2017b). A Holistic View on Academic Wide Data through Learning Analytics Dashboards. In *Proc. of the online, open and flexible higher education conference 2017 (EADTU annual conference 2017)* pp. 12–27.

Guo, P.J., Reinecke, K. (2014). Demographic differences in how students navigate through moocs. In *Proceedings of the first ACM conference on learning@ scale conference (L@S 2014)* 21–30. New York: ACM.

Ho, A., Chuang, I., Reich, J., Coleman, C., Whitehall, J., Northcutt, C., Williams, J., Hansen, J., Lopez, G., & Peterson, R. (2015). *HarvardX and MITx: Two years of open online courses*. Cambridge: HarvardX.

Jordan, K. (2014). Initial trends in enrolment and completion of massive open online courses. *The International Review of Research in Open and Distance Learning, 15*(1), 133–160.

Lamprou, A., Repenning, A., and Escherle, N. (2017). The Solothurn project — Bringing computer science education to primary schools in Switzerland. In *Proceedings of the 2017 ACM conference on innovation and technology in computer science education (ITiCSE '17)* 218–223. New York: ACM.

Liyanagunawardena, T. R., Adams, A. A., & Williams, S. A. (2013). MOOCs: A systematic study of the published literature 2008-2012. *International Review of Research in Open and Distance Learning, 14*(3), 202–227.

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.

Onah, Daniel F. O., Sinclair, J., and Boyatt, R. (2014). Dropout rates of massive open online courses: Behavioural patterns. In *Proceedings of the 6th international conference on education and new learning technologies, barcelona (EDULEARN14)* 5825–5834. Spain.

Rodger, S. H., Hayes, J., Lezin, G., Qin, H., Nelson, D., Tucker, R., Lopez, M., Cooper, S., Dann, W., & Slater, D. (2009). Engaging middle school teachers and students with Alice in a diverse set of subjects. *SIGCSE Bull., 41*(1), 271–275.

Sáez-López, J., 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.

Weintrop, D. and Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children* 199–208. Boston.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(1), 33–35. https://doi.org/10.1145/1118178.1118215.

Zheng, S. Rosson, M.B., Shih, P.C., and J.M. Carroll. (2015). Understanding student motivation, behaviors and perceptions in MOOCs. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing (CSCW'15)* 1882–1895. New York: ACM.