British Journal of Education, Society & Behavioural Science
20(3): 1-11, 2017; Article no.BJESBS.33520
ISSN: 2278-0998

SCIENCEDOMAIN *international*
www.sciencedomain.org

Redefining the Framework for Teaching Programming to Primary School Students: Results from Three Pilot Projects

Emmanuel Fokides[1*] and Pinelopi Atsikpasi[1]

[1]University of the Aegean, Rhodes, Greece.

Authors' contributions

This work was carried out in collaboration between both authors. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJESBS/2017/33520
Editor(s):
(1) Vlasta Hus, Department of Elementary Teacher Education, University of Maribor, Slovenia.
Reviewers:
(1) Stamatios Papadakis, University of Crete, Greece.
(2) Ludgleydson Fernandes De Araújo, Universidade Federal do Piauí, Brasil.
Complete Peer review History: http://www.sciencedomain.org/review-history/18968

Original Research Article

Received 18th April 2017
Accepted 4th May 2017
Published 8th May 2017

ABSTRACT

Aims: The study summarizes the findings of three pilot projects in which 2nd, 5th, and 6th-grade primary school students were taught basic programming concepts using game-like applications.
Study Design: Experimental study with one experimental and two control groups in each pilot project.
Place and Duration of Study: Sample: A total of 198 2nd, 5th, and 6th-grade students participated to the studies, coming from 3 primary schools located in Athens, Greece. The duration of the projects was between September 2016 and February 2017.
Methodology: In all projects, three groups of students were formed. One was taught using tablets or a game development programming environment, while the other two were taught the same subjects using conventional methods.
Results: Results' analyses revealed that students who used the applications had better learning outcomes, compared to the ones that were taught conventionally. The results can be attributed to the increased students' motivation and to the applications' game-like characteristics.
Conclusion: On the basis of the results, suggestions for redefining the framework for teaching programming are presented.

**Corresponding author: E-mail: fokides@aegean.gr;*

Keywords: Kodable; Kodu; mobile apps; primary school students; programming; tablets.

1. INTRODUCTION

Technology has changed many aspects of our lives. As far as education is concerned, technology has imposed a significant shift in focus; from knowledge acquisition to the acquisition of a set of skills that will render students creative and capable of responding to the needs of modern society. Students have to stop being passive users of devices and applications and become content designers and creators [1]. Even if Prensky [2] describes young students as "digital natives", as a result of their familiarity with technology, their skills are still associated with the simple use of devices and applications. The prevailing educational model continues to be that of facilitating learning through the use of technology, that is also related to the simple use of ICT tools during teaching.

The question that emerges is how we can turn students from adept users to skillful content designers and creators through technology? There are many who believe that this can be achieved if students acquire programming knowledge and skills [3]. There are multiple benefits for students when they learn how to program: development of analytical thinking, development of skills related to the design of algorithms, and a positive impact on their creativity and imagination [4,5]. Researchers suggested that when the teaching of programming becomes an enjoyable experience the results are noteworthy [6].

The teaching of programming concepts, in Greek primary schools, is included in the last two grades, not as an independent course, but a part of the IT course [7]. However, the content is poor, outdated, not well organized, and students face difficulties [8]. Therefore, a two-fold intervention is needed to rectify the problem. The first is to redefine the objectives and the content of programming as a teaching subject. The second is to find easy and fun to use tools, so as students to become motivated to learn how to program and to develop positive attitudes towards this subject.

Three pilot projects were designed and implemented over a school year. Though the target groups were students of different ages, they shared some common features: (a) the tools that were used exploited the elements of fun and play, (b) the programming concepts that were taught were basic but went beyond those included in the official curriculum, and (c) the duration and sample sizes were sufficient so that reliable conclusions can be drawn. The main research questions were: (a) to what extent primary school students can understand basic programming concepts, (b) what is the appropriate teaching method, (c) how important are the elements of fun and play, and (d) what is the role of students' autonomy during the learning process. The coming sections summarize the rationale, methodology, and findings of these projects. On the basis of the experience gained, specific suggestions on how to improve the current situation are also presented.

2. PROGRAMMING AS A TEACHING SUBJECT

Programming, as a teaching subject, is included in the Greek primary school's curriculum, as part of the Informatics' course, which is taught just for an hour each week, and only in the last two grades. Its objectives are, through the use of a simple programming language (Logo-like), students to learn how to use simple commands in order to create shapes or solve simple problems, understand algorithmic structures, and develop their problem-solving skills [7]. Apart from the fact that Logo is outdated, compared to other modern programming languages addressed to children, the curriculum is poor both in terms of its duration and content [9]. In general, students face some major problems when they learn how to program. They have a poor understanding of how programs are executed [10], and of the rules, logic, and syntax of the programming languages [11]. Variables, as well as other concepts, are not easy to grasp [12]. The reasons for the above issues are young students' lack of logical reasoning and their undeveloped algorithmic and critical thinking [13].

The teaching/learning of programming fosters a series of mental and cognitive skills. Besides learning fundamental programming concepts [14], students can develop a positive attitude toward learning computing in general [4]. A better understanding of mathematical concepts and improvement of their social skills [4], problem-solving skills [15], computational thinking [16], higher order thinking skills [17], as well as an impact on their creativity and imagination [5], were noted. There is extensive literature on the

ways that programming can be taught to primary school students. For example, Scratch and its versions attracted the attention of the scientific community [18]. Many have pointed out that its effectiveness is the result of its game-like characteristics [18]. Furthermore, very positive results yield programming environments that their purpose is the development of games. Besides having the positive effects that were previously mentioned, research has shown that such programming environments render students more creative and more motivated for learning how to program computers [19].

3. THE PILOT PROJECTS

Assuming that game-like programming environments are particularly effective, the interest turned to them and it was decided to examine their effectiveness. Therefore, in 2016, three pilot projects were designed and implemented. In the sections that follow, a brief summary of their rationale and methodology, as well as their main conclusions are presented. It has to be noted that prior to the beginning of all projects, students' parents were informed about their goals and methods and their written consent for their children's participation was obtained.

3.1 Pilot Project 1

In this project, the target group was sixth-grade students (ages 11-12). The tool chosen for teaching programming was Microsoft's Kodu (http://www.kodugamelab.com/), that allows the rapid development of 3D games. The programming language has very simple rules and it is based on physical terms and concepts such as see, hear, and bump, for the control of the games' characters and objects. Even 10-year-olds developed their own games. The programming concepts that were taught were variables, sequences, and subroutines. Two two-hour sessions were allocated for the teaching of each programming concept. Students worked in pairs. The teacher introduced each programming concept and students were then asked to developed mini-games using the programming concept that has been introduced to them.

3.1.1 Methods

To enable comparison of the results, two more groups of students were formed. To the first, only evaluation sheets, presented in the coming paragraph, were administered. Thus, it was examined what students can intuitively perceive regarding the above programming concepts. The second group was taught conventionally. The teacher taught using notes, presentations, brochures, and the whiteboard. Instead of developing mini-games, the teacher posed problems, derived from students' everyday life, associated with each programming concept, and students (working in pairs) solved them, on paper, in the form of pseudocode. For example, in sequences, they were asked to write down the recipe for a pizza in a form of a sequence of events.

The assessment of students' performance was done using: (a) evaluation sheets that were given immediately after the end of each session and (b) delayed post-tests, that were given about two weeks after the end of the project, to test the sustainability of knowledge. Each of the above tests consisted of two distinct sections. The first had multiple choice, fill-in-the-blanks, and right-wrong questions (at least 10 of them). In the second part, students were instructed to transcribe, using programming terms and concepts, everyday life activities (at least 5 such problems). Also, at the end of the project, a short questionnaire was administered in order to investigate the attitudes and opinions of students for Kodu (15 Likert-type questions). A total of 66 students participated in this project coming from three neighboring schools in Athens, Greece.

3.1.2 Results

The sample (66 students), was divided into three groups (no teaching-Group0, conventional teaching-Group1, and with the use of Kodu-Group2). For the analysis of the results, scores on the basis of the number of correct answers in each evaluation sheet were computed. Mean scores per group of participants and per test are presented in Table 1.

Table 1. Means and standard deviations on all evaluation sheets

Test	Group0 (*N* = 22)		Group1 (*N* = 22)		Group2 (*N* = 22)	
	M	***SD***	***M***	***SD***	***M***	***SD***
ES1	7.58	1.15	12.65	1.28	15.20	1.18
ES2	6.40	1.25	11.58	1.45	15.05	1.68
ES3	8.32	1.48	12.65	1.12	14.56	1.88
Delayed post-test	11.35	2.20	16.38	1.98	19.22	1.76

One-way ANOVA tests were to be conducted to compare the scores of the three groups in all tests, in order to determine if they had any significant differences. Prior to conducting these tests, it was checked whether the assumptions of ANOVA testing were violated. It was found that: (a) all sub-groups had the same number of participants, (b) there were no outliers, (c) the data were normally distributed in all tests, and (d) the homogeneity of variance was not violated, as assessed by Levene's test of homogeneity of variance. Since all assumptions for ANOVA testing were met, the analysis was conducted. The analysis showed that the teaching method had a significant effect on the scores in all tests, as presented in Table 2.

Table 2. One-way ANOVA results

Test	Result
ES1 (variables)	$F(2, 63) = 228.10$, $p < .001$
ES2 (sequences)	$F(2, 63) = 192.78$, $p < .001$
ES3 (sub-routines)	$F(2, 63) = 96.67$, $p < .001$
Delayed post-test	$F(2, 63) = 88.41$, $p < .001$

Post-hoc comparisons were conducted using the Tuckey HSD test on all possible pairwise contrasts in all tests. It was found that:

- ES1. The mean total score for Group2 (M = 15.20, SD = 1.18) was significantly higher ($p < .001$) than that of Group1 (M = 12.65, SD = 1.28), while both were significantly higher than that of Group0 (M = 7.58, SD = 1.15) ($p < .001$ in both cases).
- ES2. The mean total score for Group2 (M = 15.05, SD = 1.68) was significantly higher ($p < .001$) than that of Group1 (M = 11.58, SD = 1.45), while both were significantly higher than that of Group0 (M = 6.40, SD = 1.25) ($p < .001$ in both cases).
- ES3. The mean total score for Group2 (M = 14.56, SD = 1.88) was significantly higher ($p < .001$) than that of Group1 (M = 12.65, SD = 1.12), while both were significantly higher than that of Group0 (M = 8.32, SD = 1.48) ($p < .001$ and $p < .001$ respectively).
- Delayed post-test. The mean total score for Group2 (M = 19.22, SD = 1.76) was significantly higher ($p < .001$) than that of Group1 (M = 16.38, SD = 1.98), while both were significantly higher than that of Group0 (M = 11.35, SD = 2.20) ($p < .001$ in both cases).

Students were highly positive regarding their experiences while using Kodu. More specifically, they liked the:

- Whole project (M = 4.32, SD = 1.15).
- Application's game-like features (objects, animation, sounds) (M = 4.28, SD = 1.32).
- The process of developing games (M = 4.22, SD = 1.54).
- Programming (M = 3.70, SD = 1.56).
- Group work (M = 3.62, SD = 1.24).

Students believed that they learned quite a lot (M = 4.22, SD = 1.55) and quite easily ($M = 4.17$, $SD = 1.14$). They also found Kodu easy to use (M = 4.12, SD = 1.18), motivational ($M = 4.08$, $SD = 1.32$) and they stated that they would like to use it in other lessons ($M = 3.95$, $SD = 0.96$). No problems were reported regarding Kodu's use, while three reported problems regarding collaboration.

3.2 Pilot Project 2

In the second project, the target group was fifth-grade students (ages 10-11). The research methodology was different than the previous one. Since conventional teaching did not produce good results, it was decided to examine different types of teaching methods, but, in all, collaboration between students played a major role. Again, three groups of students were formed. All used Kodu and students worked in groups. In the first, the teacher had an active role, systematically teaching the programming concepts, by giving examples in Kodu, and by providing constant support to students. In the second, the teacher had only a supporting role (e.g., answering technical questions) and students studied the programming concepts using detailed notes. In the third group, the role of the teacher was, again limited, and the notes were not available to students; they had to seek by themselves solutions to the problems they faced.

3.2.1 Methods

The main goal was for students to develop a complex game, by the end of the project. This was implemented in three stages. First, students were asked to develop simple games, without any programming, in order to explore the objects included in Kodu. The second stage involved the development of a simple game, by adding interactions and by implementing a simple game

scenario that was given to them. Students came into contact with important programming concepts such as variables, sequences, logical expressions (AND, OR), conditions (When-Do), and subroutines. In the final stage, a detailed game scenario was given to students and they were asked to implement it in the best way they could. This stage was significantly longer, compared to the previous stages. The project lasted for about three months (70 hours for each group, 6 hours per week), due to the complexity of the tasks together with the need to provide students enough time to understand all the programming concepts and to be able to apply them. The target group was 63 fifth-grade students coming from the same schools as in the previous project.

Research data was collected by evaluating students' games. For their evaluation, the technique of content analysis was utilized (conducted by three independent raters) and a complex scoring system was developed, containing both quantitative and qualitative criteria. The quantitative criteria included the number and types of commands used, if they were used properly, and if there were any programming errors. The qualitative criteria were those proposed by Consalvo and Dutton [20]; for example, the aesthetic integrity of the game, the complexity of the levels, the complexity of commands, the gameplay, etc. In addition, at the end of the project, a short questionnaire was administered in order to examine the attitudes and opinions of students regarding Kodu (15 Likert-type questions).

3.2.2 Results

The final sample size (63 students), was divided into three groups (with teacher's assistance-Group0, with the use of notes-Group1, and with no notes and no teacher's assistance-Group2). For the analysis of the results, the three games that students developed were evaluated, with the method mentioned in the previous section and a total score for each was computed. Mean scores per group of participants and per test are presented in Table 3.

One-way ANOVA tests were to be conducted to compare the scores of the three groups in all games, in order to determine if they had any significant differences. Prior to conducting these tests, it was checked whether the assumptions of ANOVA testing were violated. It was found that: (a) all groups had the same number of participants, (b) there were no outliers, (c) the data were normally distributed in all tests, and (d) the homogeneity of variance was not violated as assessed by Levene's test of homogeneity of variance. Since all the assumptions were met, the ANOVA testing was conducted. The analyses showed that the teaching method did not have a significant effect on the scores of all games, as presented in Table 4.

Students made positive remarks regarding their experiences while using Kodu. More specifically, they liked the:

- Application's game-like features (objects, animation, sounds) (M = 4.35, SD = 1.18).
- Whole project (M = 4.23, SD = 1.23).
- The process of developing games (M = 4.05, SD = 1.24).
- Group work (M = 3.88, SD = 1.05).
- Programming (M = 3.82, SD = 1.15).

According to students' responses they learned quite a lot (M = 4.12, SD = 1.16) and quite easily (M = 4.10, SD = 1.02). They also found Kodu easy to use (M = 3.95, SD = 1.05), motivational (M = 4.18, SD = 1.15) and they stated that they would like to use it in other lessons (M = 4.12, SD = 0.85). Five students reported problems when using Kodu, while two reported problems regarding collaboration.

3.3 Pilot Project 3

The last study examined if it is possible to teach programming to younger ages that the official

Table 3. Means and standard deviations of games' evaluations

Test	Group0 (N = 21)		Group1 (N = 21)		Group2 (N = 21)	
	M	***SD***	***M***	***SD***	***M***	***SD***
Game 1	44.15	19.12	46.18	15.50	45.85	14.28
Game 2	105.10	12.15	100.22	12.58	103.85	12.14
Game 3	164.05	24.12	160.50	22.40	162.55	22.38

curriculum dictates. The target group was second-grade students (ages 7-8). Because Kodu could not be used by children of this age, tablets and an application, namely Kodable (https://www.kodable.com/), were used. Kodable was selected because of the simplicity of its interface, the game-like features, and the existence of ready-made and detailed lesson plans. Although it is in English, the interface can be easily understood rendering knowledge of English unnecessary. The student/user guides the application's character through labyrinthine levels, collecting as many coins as possible. Each level is completed when the character reaches the exit. The guidance is done by using the available commands as many times as the user wants. The commands are placed by dragging and dropping them to a limited number of empty slots, suggesting that the program must be completed using a limited number of commands. The user executes the program, sees the results, and, in case of an error, he/she can redo the programming. The levels are of escalating difficulty (e.g., more complex paths, fewer available commands). It is worth noting that there is no single correct solution to each level. Sequences, conditions (if/then) and loops were taught using this application. The lessons' plans and activities were translated and adapted into Greek.

Table 4. One-way ANOVA results

Game	Result
Game 1	$F(2, 60) = 1.15$, p = .586
Game 2	$F(2, 60) = 3.24$, $p = .602$
Game 3	$F(2, 60) = 2.85$, $p = .498$

3.3.1 Methods

At the beginning of each session, the teacher made a short introduction about the programming concept that he was about to teach, drawing examples from students' everyday lives. Next, students worked, in pairs, using the tablets, resolving the levels of the corresponding concept. In-classroom activities followed, which required teamwork and included worksheets and games. Each session lasted for two teaching hours and each programming concept required two sessions. Immediately following the end of the teaching of a programming concept, students completed an evaluation sheet, consisting of three distinct parts. The first one had multiple choice, fill-in-the-blanks, and right-wrong questions. In the second part, students were instructed to transcribe, using programming terms and concepts, everyday life activities (as in the first pilot project). The third part followed Kodable's philosophy and presentation layout. Students were presented with a level and they had either to complete the missing commands or to check whether the solution was correct (identifying any errors). Also, about a month after the end of the project, students completed a delayed post-test which had the same structure as the evaluation sheets but included all the programming concepts that they were taught. They also completed a short questionnaire for the evaluation of their experiences and views regarding the use of tablets/application (15 Likert-type questions).

For examining the significance of the project's results, two more groups of students were formed. The first one used board games instead of tablets. This method has been used by other researchers with noteworthy results [21]. Each board game was a printed and enlarged Kodable's level. The same was done for the characters and for all the other elements included in the application. The students, working in pairs, placed the various elements/commands on the board and the teacher "executed" the "program" determining if it "worked" properly. The in-classroom activities, as well as the way students worked, were the same as in the tablets group. The second group of students was taught conventionally, using notes. These notes followed Kodable's philosophy and way of presenting the learning material. Once again, students worked in pairs. The in-classroom activities were also the same as in the previous groups.

Table 5. Means and standard deviations on all evaluation sheets

Test	Group0 (N = 23)		Group1 (N = 23)		Group2 (N = 23)	
	M	***SD***	***M***	***SD***	***M***	***SD***
ES1 (21)	12.48	2.42	15.35	1.85	17.88	1.34
ES2 (20)	6.54	1.68	9.85	1.48	10.56	1.57
ES3 (20)	9.56	1.52	11.38	2.12	13.46	1.70
Delayed post-test (22)	10.13	2.45	13.58	2.91	16.85	2.17

Notes. Maximum scores for each test are reported in parenthesis. ES1 = evaluation sheet sequences, ES2 = evaluation sheet conditions, ES3 = evaluation sheet loops

3.3.2 Results

The sample size (69 students), was divided into three groups (conventional-Group0, board game-Group1, and tablets-Group2). As in the first pilot project, scores on the basis of the number of correct answers in each evaluation sheet were computed. Mean scores per group of participants and per test are presented in Table 5.

One-way ANOVA tests were to be conducted to compare the scores of the three groups in all tests, in order to determine if they had any significant differences. Prior to conducting these tests, it was checked whether the assumptions of ANOVA testing were violated. It was found that: (a) all sub-groups had the same number of participants, (b) there were no outliers, (c) the data were normally distributed in all tests, and (d) the homogeneity of variance was violated in one test, as assessed by Levene's test of homogeneity of variance. In the cases where all assumptions for ANOVA testing were met, this analysis was conducted. In the case where the assumption of the homogeneity of variance was violated, the Brown-Forsythe test was conducted, which is robust in cases of heteroscedasticity. The analyses showed that the teaching method had a significant effect on the scores in all tests, as presented in Table 6.

Table 6. One-way ANOVA results

Test	Result
ES1	Brown-Forsythe $F(2, 51.45) = 35.78$, $p < .001$
ES2	$F(2, 66) = 42.47$, $p < .001$
ES3	$F(2, 66) = 27.10$, $p < .001$
Delayed post-test	$F(2, 66) = 40.63$, $p < .001$

Post-hoc comparisons were conducted using the Tuckey HSD test on all possible pairwise contrasts in all tests except the one in which the homogeneity of variance was violated. To that, the Games-Howell test was conducted. It was found that:

- ES1. The mean total score for Group2 (M = 17.88, SD = 1.34) was significantly higher ($p < .001$) than that of Group1 (M = 15.35, SD = 1.85), while both were significantly higher than that of Group0 (M = 12.48, SD = 2.42) ($p < .001$ in both cases).
- ES2. The mean total score for Group2 (M = 10.56, SD = 1.57) was not significantly higher ($p = .286$) than that of Group1 (M = 9.85, SD = 1.48), while both were significantly higher than that of Group0 (M = 6.54, SD = 1.68) ($p < .001$ in both cases).
- ES3. The mean total score for Group2 (M = 13.46, SD = 1.70) was significantly higher ($p < .001$) than that of Group1 (M = 11.38, SD = 2.2), while both were significantly higher than that of Group0 (M = 9.56, SD = 1.52) ($p = .003$ and $p < .001$ respectively).
- Delayed post-test. The mean total score for Group2 (M = 16.85, SD = 2.17) was significantly higher ($p < .001$) than that of Group1 (M = 13.58, SD = 2.91), while both were significantly higher than that of Group0 (M = 10.13, SD = 2.45) ($p < .001$ in both cases).

Students made positive remarks regarding their experiences while using the tablets and the application. More specifically, they liked the:

- Application's game-like features (M = 4.55, SD = 1.10).
- Use of tablets (M = 4.32, SD = 1.04).
- The whole project (application and in-classroom activities) (M = 4.18, SD = 1.12).
- Group work (in-classroom activities) (M = 3.92, SD = 1.00).
- Working in pairs (application) (M = 3.86, SD = 1.12).

According to students' responses, conditions was the most interesting programming concept, followed by sequences and loops. At the same time, conditions were considered the most difficult one, followed by loops, while sequences were the easiest programming concept. In addition, students stated that they learned quite a lot (M = 4.22, SD = 1.06) and quite easily (M = 4.15, SD = 0.82). They also found tablets easy to use (M = 4.05, SD = 1.15), motivational (M = 4.00, SD = 1.05) and they stated that they would like to use them in other lessons (M = 4.19, SD = 0.65). Only one student reported problems when using tablets, while none reported problems regarding collaboration or when working in pairs.

4. DISCUSSION-TOWARDS A NEW FRAMEWORK FOR TEACHING PROGRAMMING TO PRIMARY SCHOOL STUDENTS

All three pilot projects provided useful insights on how we can teach programming to students.

Indeed, by summarizing the findings of the three pilot projects it can be noted that:

- In Pilot study 1, the results suggest that students who used Kodu outperformed students in the other two groups in all tests, including the delayed post-test. Thus, it can be argued that teaching programming through the development of digital games is a quite effective method.
- These results of Pilot study 2 suggest that students' games, regardless of the teaching methodology, did not have any statistically significant differences. This is an interesting finding, since it suggests that the teaching method did not have any statistically significant impact on students' scores.
- Pilot study 3 was different from the previous ones, since tablets were used and the target group was very young students. Taken together, the results of this study suggest that students who used the tablets/application outperformed students in the other two groups in three out of four tests, including the delayed post-test. In ES2 (conditions) the results of Group2 and Group1 were not statistically significantly different, although both outperformed students in Group0. It has to be noted that in this test the mean scores of all groups were significantly lower compared to other tests (Table 5). Indeed, by taking a closer look at this test, it was found that most students (in all groups) failed to transcribe, using programming terms and concepts, everyday life activities and also failed to complete the missing commands or to check whether the solution given to them was correct. Very few students (10 out 69) managed to complete the exercises where nested ifs should have been used.

Regarding Kodu, an important finding was that the results of the first study were in line with the findings of similar studies that indicated that Kodu made the teaching of programming concepts more enjoyable, and, because of its game-like features, it helped students to have a better understanding of basic programming concepts, and solve complex programming problems [22, 23]. After all, Kodu's main purpose is to develop games and games are compatible with children's mentality [2]. It should also be noted that students, although young, did not face significant problems while using it. Based on these findings, it can be argued that Kodu is an attractive and effective tool for teaching programming concepts to students. The third pilot demonstrated that the teaching of programming concepts, to very young students, using tablets and game-like applications, is more effective than conventional teaching methods. The findings confirmed the results of previous studies that indicated the relationship between the use of mobile devices and the good learning outcomes regarding programming concepts [24]. The absence of usage problems was noted in other studies, which attributed this finding to the familiarization of -even very young- children with electronic devices [25].

As for the appropriate teaching method, one should take into consideration the results of the second pilot. It seems that the teaching method is not so important if students have enough time to study and practice. This is supported by the fact that all groups had the same learning outcomes. This finding may seem surprising and perhaps difficult to interpret. Additionally, from the literature review, no similar methodological approaches were identified that would have allowed the comparison of the findings. But a closer examination of the results can lead to an interesting conclusion. Unlike other studies, the games, developed by the students, were examined. Therefore, what was evaluated was not an "instantaneous" effect, like in a test or in an evaluation sheet, but the result of many hours of work, trials and errors, testing, and exploring alternative solutions. It is quite possible that, initially, the three groups had differences, but these were eliminated as students had enough time to improve their games. So, even if a teaching method was not that effective, students (and their work) were the factor that balanced the results.

Consequently, one has to reflect on how the students worked. The dominant element, in all three pilots, was students' collaborative work. By, applying the constructivist views for the learning process [26], students expressed and discussed their views, and collaborated with each other [27]. Further, they had the opportunity to actively participate in the learning process, study the subject in-depth, and discover its basic principles, as suggested by the inquiry-based instruction [28]. Important elements in this view are intuitive thinking, logical leaps, originality, and the conception of radical solutions to problematic situations.

The results noted can be attributed to the use of tools that raised students' interest. Indeed, this is evident in their answers to the relevant questions. This finding is common to many studies [22,23,25]. This seems to have led to increased incentives for learning and to a better understanding of the programming concepts, which, in turn, led to better learning outcomes, as noted by other researchers [29]. Students had the ability to control the outcomes of their work and could easily monitor their progress, either by running their games in Kodu or by executing their programs in Kodable. Thus, they had greater control over the learning process and greater autonomy, as West [30] pointed out.

On the basis of the above, education administrators and policy makers can consider:

- The incorporation of game-like programming environments, such as Kodu and Kodable, into the curriculum in order to improve the way that programming is taught to primary school students.
- A teaching framework can be derived from an analysis of the methodology applied to the pilots: (a) students' collaboration and (b) with increased autonomy so as students to have the opportunity to discover, by themselves, their own solutions to specific programming problems.
- On the basis of the results, it can be argued that programming can be taught at a very early age.
- Programming courses should have enough time allocated to them (in terms of teaching hours), so the necessary skills can be developed.
- Finally, a greater involvement of teachers in the whole process should also be considered. Training will probably be necessary, but this is not expected to be that difficult as the proposed programming environments are simple to use and easy to learn.

5. CONCLUSION

This study summarized the findings of three research pilots that resulted from the need to examine the effects of using game-like programming environments in order to teach basic programming concepts to primary school students. Despite the positive results, there are limitations that need to be acknowledged. Although the samples were sufficient for statistical analysis, they were relatively small; thus, the results cannot be easily generalized. The inclusion of more programming concepts would have allowed a deeper understanding of the problem. Finally, students may not have been completely honest in their responses, confusing the questionnaires with some form of evaluation. Future studies could utilize larger sample sizes and include additional programming concepts. In order to have a wider range of results, both quantitative and qualitative methods (such as interviews with students and teachers and observations) can be used. The use of other programming tools would allow their comparison and could lead to the selection of other appropriate environments. Finally, it would be interesting to examine the learning outcomes when teaching programming to even younger ages.

In conclusion, it can hardly be said that the subject is closed. More extensive projects, in terms of duration and content, but also with the use of other tools and teaching methods, are planned for the near future. However, the evidence, so far, supports the view that game-like programming environments have a positive impact on the learning of programming concepts, especially at younger ages.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Organisation for Economic Co-operation and Development-OECD. Students, computers and learning: Making the connection. Paris: PISA, OECD Publishing; 2015.
2. Prensky M. Digital natives, digital immigrants part 1. On the Horizon. 2001;9(5):1-6.
3. Resnick M, Maloney J, Monroy-Hernández A, Rusk N, Eastmond E, Brennan K, Kafai Y. Scratch: Programming for all. Communications of the ACM. 2009;52(11):60-67.
4. Fessakis G, Gouli E, Mavroudi E. Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. Computers & Education. 2013;63:87-97.
5. Papadakis S, Kalogiannakis M, Zaranis N. Developing fundamental programming concepts and computational thinking with

scratch Jr in preschool education: A case study. International Journal of Mobile Learning and Organisation. 2016;10(3): 187-202.

6. Margulieux LE, Guzdial M, Catrambone R. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. Proceedings of the Ninth Annual International Conference on International Computing Education Research. 2012;71-78.
7. Hellenic Ministry of Education. Unified curricular framework; 2003. Available:http://www.pi-schools.gr/programs/depps/ (Accessed 10 April 2017)
8. Papadakis ST, Kalogiannakis M, Orfanakis V, Zaranis N. Novice programming environments. Scratch & App Inventor: A first comparison. In H. M Fardoun and J A. Gallud (Eds.) Proceedings of the Workshop on Interaction Design in Educational Environments, New York: ACM. 2014;1-7.
9. Grigoriadou M, Gogoulou A, Gouli E. Alternative instructional approaches in introductory programming courses: Teaching suggestions. Proceedings of the 3rd Conference on ICT in Education. 2002;239-248.
10. Pea RD. Language-independent conceptual "bugs" in novice programming. Journal of Educational Computing Research. 1986;2(1):25-36.
11. Kristi AM. Problems in learning and teaching programming-a literature study for developing visualizations in the Codewitz-Minerva Project. Codewitz Need Analysis, Institute of Software System. Finland: Tampere University of Technology. 2003;1-12.
12. Pane J, Myers B. Usability issues in the design of novice programming systems. Technical Report (CMU-CS-96-132), School of Computer Science, Carnegie Mellon University; 1996.
13. Robin A, Rountree J, Rountree N. Learning and teaching programming: A review and discussion. Computer Science Education. 2003;13(2):137-172.
14. Zhang JX, Liu L, Ordóñez de Pablos P, She J. The auxiliary role of information technology in teaching: Enhancing programming course using Alice. International Journal of Engineering Education. 2014;30(3):560-565.
15. Akcaoglu M, Koehler MJ. Cognitive outcomes from the game-design and learning (GDL) after-school program. Computers & Education. 2014;75:72-81.
16. Grover S, Pea R. Computational thinking in K-12, a review of the state of the field. Educational Researcher. 2013;42(1):38-39.
17. Kafai YB, Burke Q, Resnick M. Connected code: Why children need to learn programming. Mit Press; 2014.
18. Su AY, Huang CS, Yang SJ, Ding TJ, Hsieh YZ. Effects of annotations and homework on learning achievement: An empirical study of Scratch programming pedagogy. Journal of Educational Technology & Society. 2015;18(4):331-343.
19. Preston J, Morrison B. Entertaining education-using games-based and service-oriented learning to improve STEM education. Transactions on Edutainment III. Berlin-Heidelberg: Springer; 2009.
20. Consalvo M, Dutton N. Game analysis: Developing a methodological toolkit for the qualitative study of games. Game Studies. 2006;6(1):1-17.
21. Mavridis A, Siribianou E, Alexogiannopoulou B. Teaching programming to kindergarten and primary school students without using a computer. Proceedings of the 9th Panhellenic Conference of ICT Educators; 2015. Greek
22. Earp J, Dagnino FM, Ott M. Learning through game making: an HCI perspective. Universal Access in Human-Computer Interaction. Universal Access to Information and Knowledge. Springer. 2014;513-524.
23. Shokouhi S, Asefi F, Sheikhi B, Tee ER. Children programming analysis; Kodu and Story-Telling. Proceedings of the 3rd International Conference on Advance Information System, E-education & Development; 2013.
24. Armoni M, Meerbaum-Salant O, Ben-Ari M. From scratch to "real" programming. ACM Transactions on Computing Education (TOCE). 2015;14(4):25.
25. Goodwin K. Use of tablet technology in the classroom. NSW Department of Education and Communities; 2012.
26. Papert S. Mindstorms: Children, computers and powerful ideas (2nd Ed.). New York: Basic Books Inc; 1993.
27. Ertmer PA, Newby TJ. Behaviorism, cognitivism, constructivism: Comparing

critical features from an instructional design perspective. Performance Improvement Quarterly. 2013;26(2):43-71.
28. Dostál J. Inquiry-based instruction: Concept, essence, importance and contribution. Olomouc: Univerzita Palackého v Olomouci; 2015.
29. Snell S, Snell-Siddle C. Mobile learning: The effects of gender and age on perceptions of the use of mobile tools. Proceedings of the Second International Conference on Informatics Engineering & Information Science. 2013;274-281.
30. West DM. Mobile learning: Transforming education, engaging students, and improving outcomes. Washington, Center for Technology Innovation at Brookings; 2013.

Peer-review history:
The peer review history for this paper can be accessed here:
http://sciencedomain.org/review-history/18968