Student-Teachers’ Abilities and Attitudes towards Scratch as a Multimedia Construction tool to Represent Physical Geography Phenomena

Artemisia Stoumpa, Dionisios Skordoulis, and Apostolia (Lia) Galani

Abstract — The aim of this paper is to describe a didactic intervention that aimed to introduce pre-service teachers to visual programming with the ultimate goal of enabling them to support their students on constructing multimedia in the context of Geography in the future. Students created multimedia software about Physical Geography topics using MIT Scratch. The way the students used scientific knowledge and did the didactic transformation in order to build their artefacts has been presented elsewhere. In this article students’ ability and attitudes on programming are examined. More specifically, we study the techniques and programming structures students use, as an indicator of what students learned. Furthermore, we study difficulties on programming and other technical difficulties they confront, as well as their attitudes towards Scratch which developed through the didactic intervention. The results are encouraging since students create software that work properly and develop a positive attitude toward Scratch.

Keywords — ICT in Physical Geography, Teaching programming to pre-service teachers, Learning programming as emergent literacy, Teaching programming in non-major computer science courses.

I. INTRODUCTION

Seymour Papert’s Constructionism theory, which adopts the learning approach of building of cognitive structures, introduces the idea that knowledge building is more successful “in a context where the learner is consciously engaged in constructing a public entity, whether it's a sandcastle on the beach or a theory of the universe” [1]. A widespread technological suggestion of constructionism is the LOGO language and LOGO like programming environments. These technological proposals have enabled students to produce visual artifacts, with the assistance of teachers who are working as inspirators as well as facilitators of the realization of students’ ideas. The artifacts can be easily changed and transformed so that students can experiment with both their ideas and the ideas of their peers, thus building new knowledge. One such contemporary programming environment is Scratch. Scratch enables the construction of interactive multimedia material like games and simulations, and it is addressed to the educational community from primary education to the University. Digital games and simulations are valuable in Geography as they create a better teaching atmosphere, which helps students fulfil the learning objectives. However, the Greek educational system is being conservative, and it rarely leaves room for change. Teachers are not always sufficiently informed about new technologies in Geography education [2].

Finally, the teaching of programming as emergent literacy has been suggested by researchers in line with the teaching of natural languages. This means that as with physical languages, students initially try to express their ideas by writing small codes (in correspondence with small texts in physical languages). Through a spiral approach they gradually gain deeper knowledge of programming concepts and structures. Later as the ideas mature in their minds, comes the theory that systematizes the experiences that have been acquired through practice [3]-[6].

The aim of this paper is to describe a didactic intervention that aimed to introduce pre-service teachers to visual programming with the ultimate goal of enabling them to support their students on constructing multimedia in the context of Geography in the future. During a series of three laboratory lessons students created didactic material for Physical Geography with Scratch, by “inventing” and using these abilities of the programming environment that were suitable to materialize their own ideas.

The way the students used scientific knowledge and did the didactic transformation in order to build their artefacts has been presented elsewhere. In this article students’ ability and attitudes on programming are examined.

More specifically, we study the techniques and programming structures students use, as an indicator of what students learned. Furthermore, we study the difficulties on programming and other technical difficulties they confront, as well as their attitudes towards Scratch which developed through the didactic intervention.

II. METHODOLOGY

A. Research Questions

Taking into account the goal of this research, the following research questions have risen:

1) What programming concepts and structures do the students manage to "discover" and use?

2) What difficulties do students report in developing applications with Scratch?

3) What are the attitudes of the students towards the use of scratch as a tool for creating teaching material but also as a teaching tool in their future course as in-service teachers.

Submitted on November 4, 2021.
Published on February 16, 2022.
Artemisia Stoumpa, Department of Primary Education, ASEL Laboratory, National and Kapodistrian University of Athens, Greece. (e-mail: artemisst@primedu.uoa.gr)
Dionisios Skordoulis, Department of Primary Education, ASEL Laboratory, National and Kapodistrian University of Athens, Greece. (e-mail: dskordoulis@primedu.uoa.gr)
Apostolia (Lia) Galani, Department of Primary Education, ASEL Laboratory, National and Kapodistrian University of Athens, Greece. (e-mail: lgalani@primedu.uoa.gr).

DOI: http://dx.doi.org/10.24018/ejeng.2021.0.CIE.2759
B. Sample – Data

The sample consisted of 17 students of the Department of Primary Education who were randomly selected (first come, first accepted) to attend a laboratory course that combines the Teaching of Geography with Digital Technologies during the spring semester of 2021. Students attended the laboratory classes continuously through the WebEX platform. Students had to deliver assignments in each unit of the course. To investigate the research questions data were drawn from:
1. The software developed by the students (delivered within the course – first deliverable);
2. The documentation of the software that was sent in document format as a companion to the software (second deliverable);
3. The notes that were recorded by the teachers (diaries) during or immediately after the lessons;
4. The emails students sent to ask for help when building their software;
5. The questionnaires that were completed by the students at the end of the course;
6. The discussion with the students during the final delivery of their portfolio.

The data for this paper were predominantly drawn from the questionnaires. However, the rest of the material was used as an adjunct to interpret students’ answers when they were not clear.

C. Didactic Intervention and Educational Software Construction by the Students

As aforementioned, our research concerns the creation of interactive teaching material by students on the subject of Physical Geography. This was done in the context of a unit of four laboratory meetings-courses. Most of the students had no previous experience in programming with any programming language in general but also not on the Scratch platform. More specifically, according to their statements in the introductory course and the discussions during the final delivery of their assignments (portfolios), only four of them had some familiarity with programming and the Scratch platform. They had acquired this familiarity, during a course in Informatics that they had attended in a previous semester. The rest either had never been involved in programming or had very little contact in High School. However, they stated that they could not write even a small piece of code.

D. The Introductory Lesson (First Laboratory Meeting)

The introductory course consisted of two parts and was implemented as a classic lecture by the teachers. Students asked for clarification when needed.

In the first part the theoretical foundation of the lesson was briefly but clearly presented:

The lesson began with a reference to Constructivism, Social Constructivism, and Constructionism educational theories; the common points the differences but also the interactions of the above currents in the context of educational practice. The creation of the LOGO language in the frame of Constructionism was introduced, and the LOGO like programming environments that followed with Scratch being one of them, maybe the most prominent was explained [7]-[13]. The development of spatial skills as a goal of Geographic Education and the potential of LOGO’s turtle geometry in the acquisition of these skills was also explained. The educational robotics that can be implemented through concurrent LOGO like environments and the concrete experiences these environments offer on the geometrical exploration and re-construction of the space by creating points, lines, and shapes in different orientations and scales was presented [14]-[16]. Since these are the basic elements for maps’ creation the potential of the above-mentioned programming environments on Geographic Education was discussed.

Finally, the possibility to explore intervene and change the space that surrounds us (environment) through educational robotics and physical computing that provides us with sensors and actuators that are easily programmed through LOGO like environments as are MIT Scratch and Scratch for Arduino was presented [13], [17], [18].

In the second part of the introductory lesson:
MIT Scratch programming environment was shown. Students downloaded and installed Scratch on their own computers with the aim of the teachers.

To resemble the first LOGO Environment and LOGO’s turtle Geometry as much as possible, students were advised to delete the Cat Sprite, create a Beetle Sprite and add the pen extension. Assembling “Motion” and “Pen” Blocks in order to draw lines was shown to the students. Afterwards, they were asked to create simple shapes on different sizes and orientations on the Scratch canvas with the use of the three basic “Motion Blocks”; “move” “turn left” “turn right” (corresponding to the basic LOGO Commands Forward, Back, Right, and Left), and the “pen up”, “pen down”, “erase all” Blocks from the “Pen Extension” (that correspond to the PenUp, PenDown, and ClearScreen turtle commands). As mentioned above students were asked to use the Beetle’s sprite instead of Cat’s sprite since it has a very clear geometry and orientation on the two-dimensional canvas space as depicted in Fig. 1.

Fig. 1. Cat’s sprite, LOGO turtle, and Beetle sprite when drawing a rectangular. Turtle’s and Beetle’s orientation is clear as they move on the 2-D space. While the Cat looks like being upside down.

DOI: http://dx.doi.org/10.24018/ejeng.2021.o.CIE.2759
Afterwards, the basic concepts and algorithmic structures, of iteration, selection, variable, and subroutine (procedure/function), were introduced with the use of the appropriate Scratch Blocks. However, due to time limitations, they were only presented by the teachers, students did not have the opportunity to try them.

Students were given as homework (their assignment to be submitted on the e-class platform in five days) a series of exercises aiming at familiarizing them through practice with the above-mentioned basic concepts and programming structures.

The correction of the students’ assignments revealed that they encountered many difficulties with programming in general. More specifically, they did not understand the role of the subroutines in coding construction, neither the concept of the variable in their vast majority.

E. Three Lessons of Discovering and Exploring the Coding on Scratch as Emergent Literacy

While the introductory lesson was implemented on the traditional way as a lecture followed by a discussion solving students’ questions, the three lessons that followed, gave students the opportunity to discover the possibilities of Scratch on their own and to assimilate the programming concepts and structures taught in the introductory course by use and practise.

In the first lesson it was explained to the students that:
(i) Their assignment would be to create educational software about an issue from Physical Geography for that and the two following laboratory meetings.
(ii) They could select any topic on Physical Geography they wished.
(iii) They would have to accompany their software with a brief documentation essay. The documentation essay should contain the scientific description and explanation of the topic described on their software, the didactic transformation this explanation underwent during the design of the educational software and a brief description of the software.
(iv) They would receive feedback on their work via e-class as usual; every assignment on the course was corrected and commented by the teachers giving them the possibility to make it perfect for the next submission. It was also explained to them that in addition to the above-mentioned feedback they could ask for help mainly on construction problems from the teachers via e-mail. Moreover, software and documentation weaknesses and constructing difficulties would also be addressed in the next two lessons.

Finally, simple interactive Scratch projects, were uploaded to the e-class platform, and presented to the students, so that they could draw constructing technics to use in their own applications. However, it was stressed that they did not have to study and learn these technics, but they could use these projects as examples to draw ideas when building their own software.

The teachers took care so that these examples contained various basic commands and structures such as iteration, control, variables, and subroutines. They also took care to be examples of interface building technics as making static or animated graphics using gif images available on the internet and using colour sensing and events such as mouse touching and mouse clicking on objects to provide interaction. Scene changing, sending, and receiving messages between objects were also explained in these examples as ways to manage synchronization.

The students expressed their fear that they would not be able to design and construct their own software in just three courses. Typical was the remark of a female student: “now, we feel as if we went straight from the kindergarten (we were drawing shapes on the last lesson), to the High School”.

Five days after each lesson, the students submitted on the e-class platform the deliverable which, as mentioned above, consisted of the Scratch project and the accompanying document (documentation essay). So, each of them submitted four versions of their work: one after each lesson and one on the end of the course when handing their “portfolio”. Portfolio was a folder containing all the assignments they had created during the semester, corrected, and made as perfect as possible according to the teachers’ feedbacks. So, hereafter we will refer to four submission phases of the deliverables.

As mentioned above in the second and third lesson the assignments (software and accompanying documents) were discussed, and difficulties were addressed in the class. Without neglecting the scientific documentation and the didactic transformation, a large part of the time was devoted to solving construction issues as the students during these sessions mainly asked for technical advice presenting the difficulties, they encountered on software building.

The whole class discussed, tried, and proposed solutions; and teachers provided the appropriate scaffolding when needed. The teachers took care so that each student receive the support they needed, as much as it was possible. However, they tried to make sure that they provide exemplary solutions that would be useful to most of the class and that these examples would be close to the students’ zone of proximal development.

During this process, students gradually discovered their potential and began to express their enthusiasm for their constructions despite the weaknesses of their projects.

F. Questionnaire, and Portfolio

At the end of the semester, the students answered a questionnaire. They self-assessed the didactic transformation they made when designing and developing, their software and the achievement of the didactic objectives they set by their software. They also described the changes they made on their work (software and documentation essays) on the four submission phases, and the programming techniques they used. Moreover, they explained the difficulties they encountered on software construction and exposed their attitudes towards Scratch as a programming and an educational tool.

Finally, during the exam period, in the context of a short presentation of the portfolio of their assignments by each student, the teachers were given the opportunity to ask clarifying questions.

III. RESULTS

The following are derived from the analysis of the questionnaires:

DOI: http://dx.doi.org/10.24018/ejeng.2021.0.CIE.2759
A. Concepts and Structures Students Use

As mentioned above, while students building their software, they intuitively discovered Scratch Blocks (corresponding to programming commands on text-based programming languages) and implemented programming concepts and structures. In the questionnaire it was chosen to keep the categorization of Scratch about these concepts, structures, and Blocks as it is depicted in Fig. 2.

B. Software–User Interaction and Programming Synchronization

1) Event Blocks

Scratch allows interaction with the user and synchronization through Events Block Category. On Table I the Event Blocks students used when building their educational software is portrayed.

<table>
<thead>
<tr>
<th>Question 19: Which of the following event Blocks did you use when creating your software?</th>
</tr>
</thead>
<tbody>
<tr>
<td>When Flag clicked</td>
</tr>
<tr>
<td>When key pressed</td>
</tr>
<tr>
<td>When this sprite clicked</td>
</tr>
<tr>
<td>When backdrop switches to ……</td>
</tr>
<tr>
<td>When loudness &gt; ……</td>
</tr>
<tr>
<td>When I receive message</td>
</tr>
<tr>
<td>Broadcast message</td>
</tr>
<tr>
<td>Broadcast message and wait</td>
</tr>
<tr>
<td>Other:</td>
</tr>
<tr>
<td>I did not used any Event Blocks at all</td>
</tr>
</tbody>
</table>

The “When Flag clicked” Block is used by everyone since this was an agreement made (i.e., all programs should start and restart by clicking on the flag). Apart from this very elementary and basic event, the vast majority (14/17) used other events as well. Only three students did not use any other event, and from their answers on Question 20 they neither used any Sensing Blocks thus effectively depriving their software of user-interface interaction.

As far as those who use at least one more of the Events Blocks were concerned, the vast majority of the students (13 out of 14) used the click on object event (“When this Sprite clicked”) to allow easy and fast interaction with the user. About a quarter of the students (4 out of 14) used the keyboard (“when key … pressed”) to interact with the user. The vast majority of the students (12 out of 14) used the background change to manage synchronization on their software. This, command allowed them to organize their works into scenes, a technique enough answered and discussed during the second and the third lessons of the three lessons dedicated to educational software building (“feedback lessons”). The majority of the students (11 out of 14) used the transmission and reception of messages (“broadcast message…” and “when I receive message…” Blocks) to control synchronization.

2) Sensing Blocks

The majority of students (13/17) used some of the Sensing Blocks to give interaction to their projects. Eight students used at least 2 different Sensing Blocks. The frequency which each of these “sensors” was used is presented on Table II.

<table>
<thead>
<tr>
<th>Question 20: When creating your software, which of the following Sensing Blocks did you use?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touching an Object (SpriteX)</td>
</tr>
<tr>
<td>Touching mouse pointer</td>
</tr>
<tr>
<td>Touching a colour</td>
</tr>
<tr>
<td>A colour touching another colour (“Color … touching color …..”)</td>
</tr>
<tr>
<td>Ask (a question) and wait</td>
</tr>
<tr>
<td>Answer (on previous question)</td>
</tr>
<tr>
<td>I did not used any Sensing Blocks at all</td>
</tr>
</tbody>
</table>
As depicted on the above table, asking for a text entry from the user ("ask ... and wait") was the most frequently used command (9/17). Only five students used some other Sensing Block for user-interface interaction as follows: One detected mouse touching on an object. Four of them detect the touching of an object (for example a magnifying glass or an actor of the "story") over another object (2) or a specific colour (3).

In one of the simple interactive Scratch projects given as examples by the teachers in the first lesson, the movement of an object (magnifying glass) on a specific colour was detected. According to the teachers’ notes, in the two feedback lessons that followed, students tried to use this example. However, most of them did not manage to find images with solid colours that would be concurrently suitable to correctly present their topics on the educational software they build. As a result, the command did not "work properly" and most of them abandoned it while some of them (4) successfully modified the idea of the example. Fig. 3 shows such an attempt. In this attempt the student initially used a background image (Stage backdrop) and tried to detect the “hero’ s” touching” on the different colours of the image when the user dragged the “hero’ s” sprite over it. On the third lesson the student modified the Scratch Project by replacing the background image with four different objects (scratch sprites).

Moreover, from the teachers’ recordings it appeared that students tried to use mouse pointer touching on objects but without success. The problem with it was that since the user let the pointer on the object for a little, Scratch Interpreter detected it repeatedly. For instance, on the example depicted in Fig. 4 the student tried to reproduce “the sound” of each planet and the sun on the mouse pointer touching. She had found these recordings on the NASA website, and she thought it would be an interesting and original information well worth to be embedded on her educational application. However, when the mouse pointer touched the objects, no sound was reproduced as Scratch Interpreter repeatedly executed the script from the beginning always “starting” to reproduce the sound and never continuing to play it.

C. Functions and Procedures
1) Creation of Functions (“Own Blocks” with input) and of Procedures (“Own Blocks” without input)

Only a small proportion (6/17) created their “own blocks” on their Scratch projects (Fig. 5). Creating subroutines was taught on the introductory lesson and the simple projects given to students as examples to draw ideas for their own projects. However, no emphasis was placed on it and no time was devoted to it in the three lessons that “supported” the development of the students’ software. Maybe this is one of the reasons why students did not use this opportunity on their software.
2) Variables

Almost 50% created variables on their Scratch projects as depicted in Fig. 6.

3) Selection and Iteration Structures

As it is presented on Tables III and IV that follow, the vast majority of students discovered the usability of these two very fundamental programming structures in their simple or more complex options. Only 5 out of the whole sample (17) did not use any selection structure and just one student did not use any iteration structure either. It is worth mentioning that many created nested if and nested repeat structures, however, this is not analysed here.
D. Difficulties the Students Encountered When Constructing Their Software

Since most of students did not have any experience of programming, they encountered a lot of difficulties when building their educational applications. The difficulties that the students mentioned in the questionnaire are presented in Fig. 8.

![Software Construction difficulties](image)

**Table III: Selection Structures Students Used**

<table>
<thead>
<tr>
<th>Question 22: Did you use any “selection structures” when creating your software? If Yes please indicate which ones.</th>
</tr>
</thead>
<tbody>
<tr>
<td>If “x” then “y”</td>
</tr>
<tr>
<td>If “x” then “y” else “z”</td>
</tr>
<tr>
<td>I did not use any “selection structures” at all</td>
</tr>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>

**Table IV: Iteration Structures Students Used**

<table>
<thead>
<tr>
<th>Question 23: Did you use iterative structures when creating your software? If Yes please indicate which ones.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeat “x” times forever</td>
</tr>
<tr>
<td>I did not use any “iteration structures” at all</td>
</tr>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>

Everyone found it difficult to build their educational software. Since they had to create their software as part of one course obligations while concurrently, they had similar works to do for other courses time was limited and they did not attempt to create their own graphics. So, as it derives from teachers’ notes on the discussions during the lessons, students tried to find free graphics on the internet. However, it turned out to be difficult as it was not enough to find nice graphics, but they had to represent the physical entities without creating misunderstandings as much as possible. This is reflected in their answers on the relevant questions on the questionnaires. So, 10 students out of 17 reported that they had difficulties to find images suitable for the educational application and 9 students reported that it was difficult to find suitable animated graphics.

As recorded on teachers’ notes, synchronisation either as sound/image synchronization either as project flow control turned out to be difficult for students so they asked and retrieved a lot of help from mates and scaffolding from teachers.

For the sound-image synchronization, one technique “invented” and broadly adapted was to cut sprites-heroes’ dialogues in small parts in order to be synchronised with sound.

In projects with many scenes flow control was difficult and became more difficult as the projects became more complex with many scenes and objects. So “actors” from previous scenes appeared and spoke suddenly on next scenes since some pieces of code (scripts) continued to be executed. However, as depicted in Fig. 8 only four students reported on the questionnaire the sound / image synchronization as a problem while 7 report flow control. Six students reported difficulties on handling sensing and other events although as is analysed on section IV most were content to use the “easier” Blocks to succeed in giving interactivity to their software. Maybe they were satisfied with the interactivity they succeeded on the “easy way”.

Although students did not report it, code debugging was a difficulty that came up as software became more complex and sophisticated. In Scratch scripts were created by connecting commands that are already written in puzzle-piece shapes called Blocks. “The blocks connect to each other vertically like a jigsaw puzzle, where each data type (hat, stack, reporter, boolean, or cap) has its own shape, and a specially shaped slot for it to be inserted into, which prevents syntax errors” [19]. However, as it derived from the teachers’ notes, in large projects that were stacked many and large scripts, the code reading, and correction became difficult both for the students themselves and for the teachers when they tried to help during the “feedback lessons”.

![Use of examples](image)

**Fig. 9. How difficulties were faced: Use of examples.**

**Fig. 10. How efficient teachers’ feedback was on programming and constructing the software.**

In order to face all the above difficulties, the students tried and used the techniques from the examples given to them by the teachers, searched for more examples on the web, and asked for help from the teachers and their classmates during
the “feedback lessons”. This is reflected in their answers to the relevant questions in the questionnaires (Fig. 9 and 10).

E. Attitudes towards the use of Scratch as a Programming Tool for Creating Teaching Material and as a Teaching Tool

1) Intention to use Scratch in their future careers

Students mentioned they will use Scratch when in-service teachers. The majority (14/17) considered that they will use it to create teaching material for students. The majority (14/17) also wrote that they will put students in the role of the programmer to create simple multimedia applications or interactive stories.

All three subjects that did not consider putting the students to the role of multimedia developers with Scratch, believe that they will use it to construct didactic material. One of them considers using it to teach programming and algorithmic-logical thinking to pupils. So, finally only two out of seventeen did not consider using scratch as a didactic tool.

The three students that did not consider that they will build didactic material with scratch believe that will use scratch as didactic tool to give pupils the opportunity to create their own stories and multimedia applications (3/3), and to teach logical thinking and programming (2 out of 3).

Other ideas students have been teaching concepts (for instance variables) in mathematics and constructing games with students collaboratively.

2) Attitudes towards enhanced possibilities of Scratch (Scratch Web)

As shown in Fig. 12, the majority created the software locally. Only seven students commented on the question asking to comment on these two possibilities. All these students that commented have created their software locally. Three of them mention that using Scratch locally gets rid of the need of connection to internet and delays. One other student commented that although she created this software locally, she uses Scratch and prefers using Scratch Web since there is no need for subscription to the platform, no need to install it to the devise, and besides it gives the opportunity to share her projects with the educational community. The difficulties of using Scratch on some mobile devices and the convenience of using Scratch either online or locally on Pcs were mentioned in other answers.

As shown on the tables that follow (Table V and VI) the majority of students (12/15) do not think about publishing their software on the Scratch online community. But this does not seem to be related to whether they created their projects locally or on the Scratch Web.

TABLE V: PRE-SERVICE TEACHERS’ INTENTION TO SHARE THEIR PROJECTS ON SCRATCH WEB COMMUNITY

<table>
<thead>
<tr>
<th>Question 28: Are you thinking of sharing the software you created with the Scratch online community?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
</tr>
</tbody>
</table>

TABLE VI: PRE-SERVICE TEACHERS’ INTENTION TO SHARE THEIR PROJECTS ON SCRATCH WEB COMMUNITY vs HOW, ONLINE, OR LOCAL, THEY CONSTRUCTED THEIR SOFTWARE

<table>
<thead>
<tr>
<th>How they created their software / thinking of sharing it on the Scratch online community</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locally / Yes</td>
</tr>
<tr>
<td>Locally / No</td>
</tr>
<tr>
<td>On Scratch Web / Yes</td>
</tr>
<tr>
<td>On Scratch Web / No</td>
</tr>
<tr>
<td>Both</td>
</tr>
</tbody>
</table>

Despite students’ reluctance to publish their work in the Scratch community, on another question, the majority of them stated that they think the ability to use Scratch online is a strong advantage of the platform (Table VII).

TABLE VII: PRE-SERVICE TEACHERS’ OPINION ABOUT THE ABILITY OF USING SCRATCH ONLINE

<table>
<thead>
<tr>
<th>Question 29: How much do you agree that the ability to use Scratch online is a significant advantage of Scratch?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolutely Agree</td>
</tr>
<tr>
<td>Strongly Agree</td>
</tr>
<tr>
<td>Moderately Agree</td>
</tr>
<tr>
<td>Somewhat Agree</td>
</tr>
<tr>
<td>Disagree</td>
</tr>
</tbody>
</table>

However, when asked to justify their opinion, from those that found it as a strong advantage (“Absolutely Agree” and “Strongly Agree”), only four pointed to the possibility of sharing in the online community. And only two of these four students stated that they are thinking of sharing their software. The one student that “Moderately Agree” although she commented that it is a useful but not essential feature of Scratch, she was thinking of sharing her software on the Scratch online community.
Most students pointed out the convenience to use Scratch online on devises without administrator rights, or on devises with insufficient disk space where installation would be difficult or impossible (6). Some others pointed out the possibility to have access on ones’ projects from any devise (3) while one student described that when her project became “too big” it was impossible to work on it locally. She could only edit it on Scratch Web (Table VIII).

TABLE VIII. EXPLANATIONS-JUSTIFICATIONS OF HOW USABLE THEY FIND THE ABILITY OF USING SCRATCH ONLINE

<table>
<thead>
<tr>
<th>Question 29B: Justifications written by those that answered, “Absolutely Agree” and “Strongly Agree” on question 29.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No need of installation on the devise</td>
</tr>
<tr>
<td>Sharing to the online community</td>
</tr>
<tr>
<td>Access to ones’ projects from different devises</td>
</tr>
<tr>
<td>Scratch locally installed cannot open big projects</td>
</tr>
<tr>
<td>Do not provide an explanation-justification</td>
</tr>
</tbody>
</table>

Finally, on an open question at the end of the questionnaire students were asked to describe their experience from creating their software. On this question three students did not fill something while fourteen students did describe their thoughts.

In the 14 answers the students described the construction of software with Scratch as a pleasant or interesting experience, pointed out the possibility it offers them to create their own applications, redefined their intention to use it in the future as in-service teachers, while half of them referred to difficulties encountered (Table IX).

Three of the students stressed the importance of algorithm implementation and programming while one referred to the possibility of developing convergent and divergent thinking through programming and software building (Table IX).

TABLE IX: CATEGORIZATION OF THE EXPERIENCE OF STUDENT - TEACHERS AS EXPRESSED THROUGH THEIR COMMENTS

<table>
<thead>
<tr>
<th>Question 34: Their experience as they express it in their own words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pleasant, interesting experience</td>
</tr>
<tr>
<td>Ability to create their own software</td>
</tr>
<tr>
<td>Intention to use Scratch in the future</td>
</tr>
<tr>
<td>Difficulties</td>
</tr>
<tr>
<td>Programming- Algorithm implementation</td>
</tr>
</tbody>
</table>

Here are some typical answers:

“It was a very pleasant and creative experience. I learned to work on something that, until now, I would not have imagined. If someone showed me my project, I would never think that I could create it. Particularly useful, however, was the help from teachers. It was quite interesting, and I think it was worth the difficulty we faced in the beginning to get this result.”

“Creating software with the scratch platform was very interesting, although it was quite difficult for me. I wish we had done more lessons on the platform, so that the result would be as good as possible. However due to the time shortage we had I believe that the result was satisfactory. I would like to study it more so that I can use it more effectively in the future.”

“In the beginning, without any previous contact with the scratch environment, I was worried that I would not be able to successfully complete my work. But the more time I spent in it, the more familiar it became to me and always having in mind, as an image, my work, I felt great relief and pride when I reached the final, desired result. Engaging in Scratch was fun, it was like trying to make a puzzle.”

IV. CONCLUSIONS – DISCUSSION

In the present article, the creation of authentic artefacts – educational software by students on issues of Physical Geography is presented.

While building their applications students “invented” basic programming structures as iteration and selection structures and used many commands (Scratch Blocks) that provide user – interface interaction to their applications. They preferred “easy and clear” commands as is “ask” and “answer” Blocks, and “when this sprite clicked” Event Block. At the same time, they found difficulty on handling events such as the mouse touching an object and sensing the touching of different colours. These events could be detected through “touching mouse pointer” and “touching color …..” Blocks but the building of some more sophisticated code and techniques is needed. As far as the second Block is concerned, maybe in a next attempt an example of detecting the touch between objects must be given instead of the example of sensing colours touching that was given in the current didactic approach.

Nikolos and Komis [20] when describing a research where students future preschool teachers build educational computer games with Scratch, comment that variables would be better to be taught independently from synchronization, since managing synchronization through broadcasting and receiving messages seems to be more convenient for students. They therefore propose to teach variables in different contexts like the problem of score keeping in games.

In current research, students “invented” and preferred to use both the change of the scenic of the stage (backdrop change) and messages exchange between objects to manage synchronization. However, about half of them needed and used variables both for synchronization and to control the properties and the behaviour of objects. This need arose mainly in the bigger and more complex projects.

The type of problems that students described via email communication and on feedback lessons are very different from the problems that usually exist in “classic” lessons and programming instruction manuals. Students described their problems referring to interface issues, synchronization, and functionality of their applications, rather than language structures or programming concepts. Similar results are reported by Fesakis and Serafeim [21] who point out that building applications with Scratch “provide a realistic framework for the introduction of concepts and techniques of programming.”

In addition to programming difficulties, finding suitable graphics proved to be a serious problem in the construction of educational software. This is difficult to be addressed in a future attempt too, as it is not enough for the graphics for an educational application to be elegant but must also represent physical entities in such a way that the creation of misconceptions is avoided as much as possible.
In Scratch, code is synthesized by dragging and stacking graphic tiles corresponding to commands. This is encouraging for students as it relieves them of learning the correct syntax of the language and helps the teachers as they have to deal only with logic and not syntax errors when reading and correcting the students’ codes. However, on big projects with many and long scripts on each sprite, the code becomes “cumbersome” and reviewing and correcting or changing it becomes difficult. This is discouraging but, in the future, it might be used as an excuse to switch to another more “flexible” text-based programming language.

Finally, despite the difficulties, the students with the support of the whole class, i.e., the students among themselves and the scaffolding the teachers provided and managed to create remarkable software in an authentic problem solving environment. At the same time, while trying to solve problems (i.e., to construct the educational software), students became familiar with basic programming concepts. Moreover, students seem satisfied with their achievements and have developed a positive attitude towards programming and the Scratch programming environment, which is in line with the findings of other researches [21]-[23].

Asking from students to program educational applications was doubly demanding. However, the results are encouraging. On their software-artefacts students represent the topics from the Physical Geography preserving the scientific correctness and achieving the right didactic transformation up to a satisfactory degree (this has been presented elsewhere), while at the same time they acquired programming techniques to make their programs work.

According to Boyer [24] placing learners in the role of a multimedia designer requires extended engagement with content, thus promoting deeper understanding of the subject matter. In the current research, educational multimedia development was the reason (the authentic problem) for the subjects to delve into the scientific description and explanation of phenomena and entities of the physical world, and to be confronted with simplification, recontextualization and conventions that they have to make in order to transform the scientific explanation to didactic material (presented elsewhere). At the same time, the need to present the evolution of phenomena and entities that belong to very large or very small time and space scales (f.i. water condensation on molecular scale and water cycle on global scale, or planetary system and the corresponding phenomena) provided the motivation to seek and become familiar with programming techniques as revealed in the results section written above.

Thus, the research will continue in the future with improvements in the didactic approach based on the current findings. For example, theoretical foundation and systematic review of basic programming concepts and structures, that was presented in the introductory lesson, could be moved in the end. So that the students first discover the programming structures and concepts and the possibilities offered by the Scratch programming environment to create authentic artefacts within the educational community (emergent literacy) and then discuss their “findings” in a more systematic way. Furthermore, some techniques - solutions to the practical problems faced by students should be designed and offered early (maybe on the examples given in the first lesson) so that students can use them from the beginning, given that their time is limited and divided into many tasks and obligations they have to fulfill for the different courses during the semester.

REFERENCES


DOI: http://dx.doi.org/10.24018/ejeng.2021.0.CIE.2759
Artemisa Stoumpa is an assistant researcher in the ASEL Laboratory, at the Primary Education Department of the National and Kapodistrian University of Athens. She holds a Bachelor's degree in Chemistry (University of Ioannina, 1991), a Master's in Education on the Informatics in Education (National and Kapodistrian University of Athens, 2004), and a PHD in ICT on Environmental Education (National and Kapodistrian University of Athens, 2011).

She has taught the course The Role of the Experiment on Natural Sciences Teaching from 2019 to 2021, as well as the course The Digital technologies on Geography Education in the academic year 2020-2021. She has taught the course Theoretical approaches on STEM Education at the Primary Education Department of the University of Ioannina in the academic year 2018-2019.

In parallel, she works as a teacher in the Second Model high school of Athens. Her main research interests include STEM Education, Science Didactics, and Educational Robotics.

Dionysios Skordoulis is an assistant researcher in the department of primary education of National and Kapodistrian University of Athens. He has graduated from the department of computer engineering from the university of Central Greece and he holds the MSC in Information Systems. He has attended the seminar of STEM Education and Educational Robotics and he has participated in multiple projects ever since.

Apostolia (Lia) Galani is Associate Professor in Didactics of Geography at the National and Kapodistrian University of Athens, Greece. She has authored or coauthored many articles or chapters in Greek and international journals or books, and she is the author-coauthor of 15 books, including the Greek primary and lower secondary education geography textbooks. Her research interests focus on socio-scientific issues, on the educational approach of environmental issues with innovative methods (e.g., remote sensing), on the use of ICT in school Geography.