Investigating the Transition from Block-based to Text-based Programming Techniques in Secondary Education in Greece

Eleni Seralidou and Christos Douligeris

Abstract — Nowadays computer programming is an essential skill that has grown in popularity in secondary education. In its block-based form, it is used to familiarize students with the basic concepts of programming before the students move on to text-based programming languages. The purpose of this paper is to investigate the transition from block-based to text-based programming techniques in secondary education. More specifically, four educational programming activities, which exploit the use of a block-based programming environment, were implemented in two classes of third grade Gymnasium students in Greece during the course of Informatics. The same educational activities were implemented again in the same classes with the use of a transitional block-to-text based programming environment aiming at normalizing the transition from block-based to text-based programming techniques. The implementation findings indicate very positive feedback from the students’ side in understanding the association between block and text commands.

Keywords — Block-based, Programming, Secondary Education, Text-based.

I. INTRODUCTION

Programming is a complicated and demanding task, which is particularly difficult for young students and novice learners to master [1]. In this context, teachers also face several challenges when presenting the fundamental concepts of programming in the classroom [2], [3]. One of the challenges is the selection of the right programming environment in order to support the learning process [4].

Furthermore, it is important to decide the teaching and learning approach in different student ages having in mind the different forms of programming. For example, programming in block-based environments is considered as a key element of introductory programming in K-12 settings, while past research conducted in the context of text-based programming points to several challenges related to the novice learners’ understanding of foundational programming constructs such as variables, loops, and expressions [5]. The use of block-based programming environments is purported to be a good way in order to gently introduce novice computer programmers to computer programming [6]. One justification for the use of block-based environments in formal educational settings is the idea that the concepts and practices developed using these introductory tools will prepare learners for future computer science learning opportunities because they are supposed to allow students to focus on concepts, leaving aside the syntactic complexity of a text-based language [7]. This view is built on the assumption that the attitudinal and conceptual learning gains made while working in the introductory block-based environments will transfer to conventional text-based programming languages [8].

These block-based environments by using blocks of code intend to familiarize students with the programming logic [9], before introducing them to text-based programming languages such as Python. They differ significantly from their text-based counterparts and have proven to be successful in motivating children and making it easy to start programming [10].

In parallel, many researchers have explored the benefits of using block-based programming environments for teaching basic programming principles in early ages with positive results. Tumlin [11] created COPPER (CustOmizable Puzzle Programming EnviRonment) which refers it as a meta-configurable tool for creating coding puzzles on a grid using Google’s Blockly framework to eliminate the mental overhead of memorizing textual syntax, allowing students to focus on building computational thinking skills. The creator argues that it has the potential to lead to higher student interest and comprehension of programming concepts in a customized context. Bers [12] proposed a different approach for computer science education in early childhood through the use of the block-based programming environment Scratch Jr. He argued that programming languages and pedagogies need to be developmentally appropriate for young children, because it is more important than ever to develop computer science pedagogies that promote a deep and thorough engagement for everyone starting in early childhood.

While block-based languages have exploded in popularity, little work has been done to show that the students’ learning in these environments is effectively transitioning to more traditional text-based languages like Java [13]. Additionally, several studies have recognized various drawbacks of these block-based environments which could potentially be detrimental when students transition to text-based languages [10]. Transitioning across the significant gap between the two editing styles presents a difficult challenge in school-level teaching of programming [14] and also for learners themselves, who can encounter difficulties when transitioning to text-based languages. Nevertheless, many learners have a strong desire to make...
that transition anyway [15].

At the same time, some research has been done for the use of hybrid programming environments that combine block-based with text-based programming techniques. More specifically, in these programming environments blocks are used as commands but also the display of commands in text form is included. Homer and Noble [16] introduced Tiled Grace, a block-based programming system backed by a conventional textual language that allows switching back and forth between block-based and textual editing of the same code at any time. They implemented this programming environment primarily in students enrolled in early undergraduate Java courses in the School of Engineering and Computer Science at Victoria University of Wellington selected so that they would have some existing familiarity with the idea of programming. The participants showed high levels of engagement. The feature of providing multiple views of code helped them to feel more comfortable when using the specific programming environment. Strong et al. [15] introduced the use of pyBlockly, a programming tool that allows users to continue to use the block-based editing style while they become familiar with the syntax and semantics of a traditional programming language. The tool allows learners to construct programs in the Python syntax, targeting the familiar Turtle Graphics microworld. This allows learners to engage with the conceptual challenges of changing language and programming style while using a familiar block-based interface. Their aim is to support learners in the early stage of moving from block-based languages such as Scratch to text-based languages and particularly Python.

Consequently, some blended and dual-modality, or else hybrid, programming environments have emerged that integrate affordances of the block-based programming approach into conventional text-based programming interfaces. Despite this growth in the variety of introductory programming environments, relatively little is known about the relationship between modality, programming interface design, and the impact they have on the learners’ emerging programming practices [17].

Considering the above, efforts have been made for bridging the gap between block-based and text-based programming techniques, but research in this field is still open. Despite of the growing use of block-based programming environments in formal settings, relatively little empirical work has been done to understand the impacts of their use in high school classrooms [18]. Furthermore, most of the hybrid programming environments need to be tested in real classroom conditions by teachers and students in order to witness their effect in the teaching and learning process. The question that needs to be answered is: can there be a programming environment that helps students understand the association of programming principles from block-based to text-based type of commands and support their transition from one programming style to the other? Through a general perspective, do block-based programming tools help learners transitioning to text-based languages?

In Greece, programming lessons are included in the course of Informatics in the secondary education curricula for all the Gymnasium (Greek Junior High School) and Lyceum (Greek Senior High School) classes. Some of the programming environments that are suggested from the Greek Institute of Educational Policy for the Gymnasium and the first grade of Lyceum are Scratch, Kodu Game Lab and AppInventor, which are all block-based programming environments. Furthermore, since the year 2000 in Greece, the course of Informatics has been taught in the third grade of Lyceum and the main purpose of the course is to enhance the students’ thinking and focus on problem solving techniques through learning text-based programming. For all these years, this course is also included in the Pan-Hellenic exams for entering tertiary education. That means that the students who reach the second grade of the Lyceum must switch from block-based to text-based programming without having a transition phase in between. Thus, it is important to know if they can understand the programming principles that they have been taught in the previous classes and use them in the new text-based programming style environments.

In this paper, we present our approach for the transition from block-based to text-based programming and, more specifically, to the Python programming language. For this purpose, we created four educational activities that were implemented in the third grade of Gymnasium with the participation of thirty five students. The same educational activities were implemented both in a block-based and in a hybrid-programming environment. After the implementation a discussion session with the students followed in order to determine if the transition was helpful and successful, from which we received positive feedback.

The rest of this paper is organized as follows. In Section II, the contemporary programming approaches are presented. In Section III, a description of the programming environments that have been exploited in the described implementation is presented followed in Section IV by the description of the educational activities and their association with the Informatics course curriculum. In Section V, the research approach of this study is presented and in Section VI the results of the discussion session with the students are included. Lastly, Section VII includes the discussion and Section VIII the conclusion and the future steps.

II. CONTEMPORARY PROGRAMMING APPROACHES

A. Block-Based Programming

Computer programming has grown in popularity over the last years. The focus is not anymore on adults but the interest in programming has flourished in younger ages as well. Because of this interesting transformation, various approaches in programming styles and techniques need to be adopted in order to help kids in early ages get accustomed with basic programming principles in an effective way. This concern led to the block commands programming approach and the design and development of block-based programming environments.

Some of the most popular block-based programming environments for education are Kodu Game Lab, Scratch, AppInventor and Blockly. These environments use blocks that can be attached to one another and create a puzzle-like form of code instead of text commands. The benefits from
using this type of programming environments, according to recent research, can be summarized to the following:

- the block-based programming environments can facilitate the process of learning programming and students misconceptions about programming principles are minimized [19],
- through the use of block-based programming environments the students in early ages can master foundational coding concepts [20], and
- through the use of block-based programming environments the students focus on programming concepts leaving aside the syntax complexity of a text-based programming language [7].

In Greece, the Informatics courses curriculum suggests that the teaching of programming be performed with the use of block-based programming environments starting from pre-school education up to the first grade of Lyceum students, suggesting the use of many different environments and allowing the educators to decide which is appropriate for their classes.

B. Text-Based Programming

Teaching and learning text-based programming languages is not an easy process due to the complex structure of their content. This fact results in various obstacles that learners must overcome during the learning process and in many challenges for educators during the teaching process [21].

In Greece, the Informatics courses curriculum proposes the teaching of the Python programming language for General and Vocational Lyceum students and the Java programming language for Vocational Lyceum students in the specialty of Informatics. Additionally, for the students of the third grade of Lyceum a pseudo-programming language, under the name “Glossomathia”, is suggested. This pseudo-language includes text commands in Greek, through which the students are able to create programs and execute them in the “Glossa” interpret program. Through this process, the students get accustomed in syntax and advanced programming principles through the creation of algorithms in Greek and they can switch to any real text-based programming language easily because the “Glossa” interpret follows the rules and syntax of structured programming languages. So, one can say that the students also learn text-based programming in Greek. In earlier ages some reference to text-based programming languages is included but with no further analysis.

C. Hybrid Programming

As it was mentioned above, research has shown that a gap still exists when it comes to the transition phase from block-based to text-based programming environments despite the fact that some efforts to investigate this transition have been made. Research on this field led to the design and creation of hybrid programming environments, which combine blocks and text commands. The benefits from using this type of programming environments can be summarized to the following:

- the hybrid-programming environments can facilitate the process of transitioning from a block-based to a text-based programming language,
- through the use of hybrid-programming environments the students can understand easily the connection between coding concepts in different programming styles,
- through the use of hybrid-programming environments the students get accustomed to programming language syntax issues while using block commands.

In Greece the Informatics courses curriculum do not suggest the use of hybrid-programming environments.

III. PROGRAMMING ENVIRONMENTS

For the purpose of this study, two programming environments were used in order to cover both block-based and text-based programming styles. For the block-based version, we used Scratch and for the transition phase to text-based programming, we used Edublocks. Scratch and Edublocks are described in the following paragraphs.

A. Scratch

Scratch is one of the most popular block-based programming environment (Fig.1). It is available on https://scratch.mit.edu/ and it can be used online, or it can be installed locally on a device. One of the main reasons of its popularity is that the users do not need to know any particular programming language, since the relevant code is developed through the use of block commands [22]. This feature makes it appropriate for use in primary and secondary education allowing learning to become more interested in programming, more challenged and more creative [23].

Additionally, for young students it is important to understand, to the greatest possible extent, the use and the results of programming structures and how these structures can be used properly in a program in order to solve a specific problem [24] and then move on to more advanced programming techniques. By using the Scratch programming environment, students can be taught basic programming concepts like conditionals, loops and variables giving emphasis to the way these structures function rather than focusing on syntax or more technical programming issues.

Fig. 1. The Scratch programming environment.

B. Edublocks

Edublocks is an online hybrid-programming environment, which includes block commands in Python (Fig. 2). It is
available on https://edublocks.org/ and supports Python v3. One great advantage of this environment is that it supports a dual command view. The blocks’ view allows the user to see the Python commands in the form of blocks and the Python view allows the user to see the full Python syntax (Fig. 3).

In addition, the feature of Split view is supported, through which the user can see both block and text commands without having to switch views (Fig. 4).

Additionally, an effort of combining the Scratch with the Python block commands is included on the Edublocks webpage. Instructions in the form of coding cards (Fig. 5) offer for basic Scratch command blocks their equivalent in Python blocks in order to help beginners understand the Edublocks commands. So far ten code examples are offered which are helpful but still not enough for a new user.

IV. THE EDUCATIONAL ACTIVITIES

A. The Informatics Course’s Curriculum

The curriculum of the course of Informatics for the third grade of Gymnasiums in Greece aims at teaching problem solving techniques and at enhancing the students’ structured thinking. It is taught for one didactic hour per week, while the teaching instructions suggest fourteen didactic hours of teaching programming in total for a school year. The course’s topics include the learning of basic programming concepts through the creation of algorithms and programs using Scratch together with other suggested software. More specifically, concepts like conditionals, loops and variables must be taught to the students through the creation of simple programs and games.

Although new concepts and teaching techniques are suggested in the teaching instructions, a great disadvantage is the lack of an up-to-date schoolbook that could support the teaching and learning process. Furthermore, there is no plan and forethought about the transition process to the text-based programming style that is included in the Lyceum classes for older students.

B. Activities’ Description

The activities were specifically designed for the course Informatics of the third grade of the Gymnasium. The focus is on the third part of the existing schoolbook, which covers the subject of programming including the creation of algorithms and visual programming.

For this purpose, we produced worksheets, which include four activities, for the creation of various programs that exploit the use of conditionals, loops, variables, and input/output commands. For the design and structure of our activities, we took under consideration relative learning theories, focusing mainly on the constructivistic approach.

More specifically, the first activity focuses on input/output commands, like ask or say in Scratch, and the simple conditional structure if-then, the second activity uses again input/output commands in combination with the more complex conditional structure of if-then-else and the use of logic operators. The third activity focuses on loops and specifically on the repeat loop in Scratch, while the fourth activity again uses loops but with an emphasis on the use of the repeat until loop structure in combination with variables.

By completing these activities, the students through creating four simple programs are taught the structures of conditionals, loops, variables and input/output commands, and also principles of object-oriented programming, which
comes in full terms with the Informatics course teaching instructions of the Greek ministry of education.

The total time planned for the use of the Scratch programming environment was four didactic hours. After the implementation and the explaining of the use of the pre-mentioned programming concepts, we moved on to the next programming environment and implemented the same activities for four more didactic hours with the use of the Edublocks programming environment, as it is described in the next section.

For additional help during the activities’ implementation, we also created our own version of coding cards, which included all the block commands that are needed for each activity in Scratch and Edublocks respectively (Table I).

| Scra
<table>
<thead>
<tr>
<th>Edublocks Blocks</th>
<th>Where to find it in the Edublocks Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch Blocks</td>
<td>Statements menu</td>
</tr>
<tr>
<td>Edublocks Blocks</td>
<td>Logic Menu</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In a discussion session that followed, the students gave their opinion about the activities and the programming environments in general. This discussion session qualitative research method was useful and necessary for understanding the meaning and context of the material studied [25], and in this case helped in gathering the opinions of students that can considerably help in improving the suggested activities.

After completing the entire set of activities and discussing and explaining any misconceptions, a final activity which combined the previous taught concepts was assigned as homework to the students with only one restriction which referred to the exclusive use of Edublocks.

A. Methodology and Evaluation

The purpose of this research was to gather the students’ opinions and suggestions over the proposed activities. The participants were students from two third grade classes of a Gymnasium located in Nikaia – Piraeus, Greece and their ages were from 14 to 15 years old. We selected this specific age range taking into account the research of Karaliopoulou et al. [1] which determined the age after 14 years as the recommended age for transitioning from block-based to text-based programming.

Initially, in the online synchronous lesson, through the webex platform, each week’s activity was presented to the students. As the activity’s subject was explained, a targeted reference in basic programming concepts was made. Concepts like the conditionals, the loops, the variables, and input/output commands were mentioned and explained thoroughly.

After that, the worksheet activities were presented and made available to the students in the asynchronous lesson environment, through the e-class platform. The students had the opportunity to work on each activity for one week, in their own time.

During the next online synchronous lesson, each student was asked to deliver their own version of the code of commands. Through the screen sharing option, the students were able to show the activity’s execution and explain the code they created using block commands. In several cases, the students decided to expand some of the exercise options by adding their own ideas.

The next step was the transition to the programming environment of Edublocks. First, the environment’s functions were explained to the students and a clear correlation between the blocks of commands in Scratch with the Python Blocks in Edublocks was made in order for the students to understand which command they need and for what purpose. Our version of the coding cards, one for each activity, was also made available to the students in order to further help them with the transition to the new programming environment.

After that, the students had once more the opportunity to work each activity for one week, in their own time and again during the next online synchronous lesson, each student was asked to deliver their own version of the code of commands.

In order to get further insights, we also asked from the participating students to share their opinions through a discussion session. This session was performed after the completion of all the activities in the transition phase, in person, and lasted a full didactic hour for each class separately.

First, a conversation was held regarding the two different programming styles, next about the activities content and
Lastly about the concept of transitioning to text-based programming. After that, more attention was paid to the quality of the content, the difficulty in executing the activities and the interest and emotion that was derived in the implementation process. The responses were noted in order to be used for reaching conclusions, as we will see in the next section.

Finally, one last activity which combined all the previous taught concepts was assigned to the students as homework in order to acquire additional information about their performance.

VI. DISCUSSION SESSION AND RESULTS

The number of students who participated in the discussion session was 35 coming from two third grade classes, from which 14 were girls and 21 were boys at the ages of 14 to 15 years. All the students had previous knowledge in programming with the Scratch environment.

The discussion session lasted a full didactic hour for each class separately. First, the teacher outlined the basic activities’ actions in order to warm up the conversation and then posed seven questions:

- What do you think about the activities’ context?
- Did the worksheets help you understand what was asked by the instructions given?
- Do you think that you had to place considerable effort in completing the suggested activities?
- What do you think about the programming environment of Edublocks?
- Do you think that you had to place more effort in completing the activities using the Edublocks programming environment than using the Scratch programming environment?
- After completing all the activities, in both programming environments, do you think that you understand better the concepts of input/output commands, conditionals, loops and variables?
- What do you think about the Python programming language?

Some of the answers received are presented in Table II.

<table>
<thead>
<tr>
<th>Questions</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Question</td>
<td>I think the activities are easy to work with.</td>
</tr>
<tr>
<td></td>
<td>Some were difficult but I figured them out.</td>
</tr>
<tr>
<td>Second Question</td>
<td>Yes, it was clear.</td>
</tr>
<tr>
<td></td>
<td>They helped me.</td>
</tr>
<tr>
<td>Third Question</td>
<td>I some of them yes.</td>
</tr>
<tr>
<td></td>
<td>No, I think they were easy.</td>
</tr>
<tr>
<td>Fourth Question</td>
<td>Seems to be very interesting.</td>
</tr>
<tr>
<td></td>
<td>I was confused at the beginning but now I think its pretty easy.</td>
</tr>
<tr>
<td>Fifth Question</td>
<td>The first time I used it I felt that I did not know anything but after that everything was easy.</td>
</tr>
<tr>
<td></td>
<td>At the beginning it seemed harder than Scratch but its not</td>
</tr>
<tr>
<td>Sixth Question</td>
<td>I think I understand these concepts pretty well.</td>
</tr>
<tr>
<td></td>
<td>I do not feel very confident, maybe because we were in distance learning all this time.</td>
</tr>
<tr>
<td>Seventh Question</td>
<td>I could never imagine that real programming looked like this!</td>
</tr>
<tr>
<td></td>
<td>Wow! Now that I learned Python I feel like a real programmer!</td>
</tr>
</tbody>
</table>

As a general result of the discussion session one can say that the students think that learning programming using the hybrid programming environment of Edublocks is interesting, easy and does not require a great amount of effort.

During the discussion session, we also asked the students to give us their opinion for the suggested activities improvement. Some of the comments included the following suggestions:

- I think that after completing everything there should be an activity entirely in the Python language.
- I think the activities could be more complicated and include even more difficult commands.

Lastly, through the process of grading the students delivered activities and also grading the final homework activity, a general improvement was observed. More specifically, 65% of the participating students improved their grades, 26.5% remained at the same grade level, while only 8.5% did not perform well.

VII. DISCUSSION

The previous results raise interesting issues. According to the opinions of the students, who used and implemented the activities, their context was simple and interesting. Also, the students found really helpful that they had in hand additional coding cards for each activity, so they did not have to place considerable effort to complete the tasks.

Furthermore, most of the students seem to like the idea of learning a “real” programming language instead of Scratch and showed a great deal of enthusiasm. Additionally, the students feel that they understand better the programming concepts they were taught and seemed more confident in using and explaining concepts like input/output commands, conditionals, loops and variables.

Lastly, the majority of the students showed a very positive attitude towards a hybrid programming environment, they suggested their own extensions and they delivered interesting and well performed versions of the activities.

VIII. CONCLUSION – FUTURE WORK

First of all, we need to mention that we received a really positive feedback by the students who participated, used and implemented the activities.

According to the students’ opinion, the activities are easy and help them to understand concepts of programming during the transition phase. Furthermore, they think that the hybrid programming environment of Edublocks is easy to use and very informative about how a “real” programming language is and works.

In the future, we intend to include more activities with different levels of difficulty and improve the existing ones where needed. Also, taking the students suggestions under account we will design an activity for the Python language that can be implemented after the transition phase.

After these improvements, we are going to ask the teachers’ opinions in order to test the improved activities’ content and then proceed in their implementation and thorough evaluation by more students in school classes.
ACKNOWLEDGMENT

This work has been supported by the Greek Secretariat of Research and Technology through a matching funds grant for project SAURON. We thank the thirty five students who offered their support through the use of the proposed activities, the participation in the discussion and the provision of suggestions for improvement.

REFERENCES


Eleni Seralidou received the BSc in Informatics from the department of Informatics of the Technological Educational Institute of Athens - Greece in 2002. She received the MSc in Advanced Information Systems in 2011 and a PhD in “Collaborative Internet Tools in Education” in 2021 from the department of Informatics of the University of Piraeus - Greece. Since May 2021 she is a PostDoc Researcher in the department of Informatics of the University of Piraeus. She currently works as an Informatics teacher in secondary education and in addition she works as an adult educator in post-secondary and postgraduate studies. Additionally, from 2014 until today she is a workshop teaching assistant of the courses “Internet Technologies” and “Networks” at the Department of Informatics of the University of Piraeus and a member of the Research Laboratory of Telecommunication Services, Systems and Security (NetLab) of the Department of Informatics of the University of Piraeus, which operates under the supervision of Professor C. Douligeris. She is also the author of several articles in international and local scientific conferences and in scientific journals from 2010 until today.

Christos Douligeris, currently a professor at the Department of Informatics, University of Piraeus, Greece held positions with the Department of Electrical and Computer Engineering at the University of Miami. He was an associate member of the Hellenic Authority for Information and Communication Assurance and Privacy and the President and CEO Hellenic Electronic Governance for Social Security SA. Dr. Douligeris has published extensively in the networking scientific literature and he has participated in many research and development projects. His main research interests lie in the areas of communications, networking security, cyber security, educational technologies, and the use of technology in all aspects of addressing immigration issues, web science, data analytics, and emergency response operations.