Object-Oriented Programming for non-IT Students: Starting from Scratch

http://dx.doi.org/10.3991/ijep.v5i4.4734

O. Mironova, I. Amitan, J. Vendelin, J. Vilipõld and M. Saar
Tallinn University of Technology, Tallinn, Estonia

***Abstract*—The present paper demonstrates a teaching approach in general programming course for the first year non-IT students at Tallinn University of Technology, Estonia. The authors suggest some ways for achieving better results in programming issues that are usually complicated for the beginners.**

***Index Terms*—object-oriented programming, Scratch, VBA, Python.**

I. INTRODUCTION

Information technology and programmable systems, such as computers, smartphones and other devices, are playing an increasingly greater role in our lives today, both at home and at work. We can see growing demand for information technology professionals and escalating need for the knowledge about computer technology for other specialists. "Computational thinking" is a skill that all students must learn to be ready for the workplace and able to participate effectively in the digital world.

Basic computer education topics have been included into all the curricula at Tallinn University of Technology, Estonia, and have been integrated into a course named "Informatics".

The Informatics lasts two semesters and number of weekly study hours is two. Group size is 20-30 students. During the course we apply classic face-to-face classroom methods, group work and learning in the Moodle e-environment.

The main learning outcomes in the Informatics course are listed below. Students who complete the course:

- Acquires the foundations of problem analysis and system modelling.
- Can analyse relations between objects and provide rationale for the algorithms and methods applied.
- Is familiar with the nature of data and objects and can specify them and use them in programs.
- Is familiar with and can describe using VBA/Python and UML activity diagrams main activities occurring in programs and algorithms.
- Is familiar with the nature and main concepts of object-oriented programming.
- Can compose programs consisting of multiple procedures and organize data flow between them.
- Can use graphical objects in programs; develop scaled drawings and schemes, movements, and animation in VBA.

The course starts with application development in the environment of general-purpose application software such as document and spreadsheet processing. The second part is devoted to building algorithms and programming. The aim of this part is to develop logical, analytical and algorithmic reasoning skills as well as the ability to investigate problems and tasks in a systematic way.

The course aims at reaching the results in two different but tightly linked ways: learning to understand the object-oriented approach in the description of different concepts and getting necessary skills in building algorithms. Both skills have to be implemented in simple applications.

It should be mentioned that the Informatics course seems to be rather sophisticated for most of the non-IT students. The main issues in teaching the subject have been delineated and systemized in [1], [2]. However, we still face some problems. Consequently, we try to improve the course content from year to year and from speciality to speciality to find the best methods to achieve the goals.

After several years' experience two main algorithmic languages were chosen for creating applications. These are Python and Visual Basic for Applications (VBA). We have different reasons why we prefer one or the other, but there are a lot of things that should be considered and mastered beforehand.

II. THE REVIEW OF THE CURRENT SITUATION IN THE COMPUTING TEACHING

In recent years, several countries have carried out thorough investigations of the use of information technology and courses on computer science in different schools. Analyses have shown that most of the courses do not meet the needs. As a result, several new curricula have been proposed to improve the situation.

In 2011 the new CS standard, "CSTA K-12 Computer Science Standards" was created [3]. It sets out the basic requirements for the various areas and levels of the curricula. A number of courses and subject syllabuses were created on this basis. One of the most outstanding is the new CS syllabus "AP Computer Science Principles" [4] created under the support of US National Science Foundation (NSF). The work started in 2011 and the course is planned to be completed in 2016.

The documents mentioned above are based on the notion of "Computational Thinking", which defines general principles for describing the problems and solving them by means of software systems, including such concepts as abstraction and modelling, algorithms, data and information, programming, communicating and collaborating. A large part of the concepts is related to algorithms and programming [5].

In 2012 a comprehensive study "Shut down or restart?" was published by The Royal Society UK [6]. The research brought out significant shortcomings and offered ways to solve them. "Computer Science: A Curriculum for Schools" [7] was set up and published, where "Computational Thinking" is the main idea. Starting from September 2014 the course Computing (Computer Science + Information Technology + Digital Literacy) is included in the UK school curricula.

In our teaching approach to the programming course we use the principles introduced above and try to implement them in the best possible ways.

The current situation of teaching computer sciences at our country schools is quite discrepant. Some schools do not have informatics lessons at all, in some schools it is taught only for two or three years, which is a very short period to prepare students for the university level. This drawback is associated with two main reasons. The first one being that there is no nationwide Informatics curriculum in our country. The second one is that Informatics subjects are not mandatory in our schools. A logical consequence of these reasons is the situation where each school teacher introduces learners to the material at his own discretion: certain pupils draw in Paint, others learn the computer hardware in theory, etc. In connection with this, the level of PC skills among non-IT learners falls every year and reduces to commonplace Facebook usage. In our course we have to take these facts into account and build the curricula accordingly.

III. MODELLING

During the Informatics course students have to create simple applications. After the task is set, we go to the next step: modelling. There are two main aspects to learn: defining the data objects and building algorithms.

The object-oriented approach is the main technique in building and developing software applications and information systems. Its essence is in describing the properties and the behaviour of real and abstract subjects by means of software objects.

The software object is a collection of connected data and programs. The computer system and application software uses object-oriented approach to define documents, user forms, windows, toolbars, etc. The newer programming languages are object-oriented, where the basic concepts are classes, properties, methods and events. The relationships between the objects are used as well.

Aspects containing the description of objects make up a significant part of the application model.

Another part of the modelling process is the description of algorithms used mostly as object methods.

The algorithm is often built step-by-step, starting with general description and becoming more specific later. Finally, we come to a level, where it is easy to transform the outcome into a program code.

UML (Unified Modelling Language) supports building models for applications with a set of diagrams. It helps the learners to reach a solution without reference to any of the programming languages. It is widely used not only in the general programming course, but also later in courses on information systems, databases and several others. However, UML diagrams provide static views and are not always the best to keep track of the process.

IV. VISUAL PROGRAMMING WITH SCRATCH

A new trend in teaching programming skills is the development of an environment created especially for learning. These are graphical tools, such as Scratch [8], Snap! [9], Blockly [10], which make the learning process much easier for the beginners especially for non-IT, who have not any experience in programming.

In our course we use Scratch as the supporting tool before creating applications in VBA or Python. After a few years of practice, we came to the conclusion that a graphical environment, such as Scratch, is an effective introductory tool to understand both the object-oriented approach and the functionality of a program.

In addition, syntax errors are impossible in Scratch, which is a great help for students. It is easy to discover and correct run-time errors as well, because Scratch works as an interpreter.

Graphical command blocks give a visual picture of the different controls (selections, loops), used in the program. They create the necessary associations when students start coding in a text-based programming language.

Scratch is not designed to solve complicated tasks, but it is simple, very expressive, and makes understanding the behaviour of objects easier.

It should be mentioned that according the annual students' feedback Scratch is the most popular module in the course.

Creation of objects is solved by importing or drawing graphics. Combining the blocks for each object creates the methods. Some of the blocks are used to show the reaction of the object to some events. We see here the main aspects of object-oriented programming resulting in an attractive animation.

Further, we will briefly review the main programming concepts that are usually complicated for non-IT learners.

A. Objects and their properties

Each Scratch object (Sprite) has properties like the name, coordinates, direction, rotation style, etc. (Fig. 1).

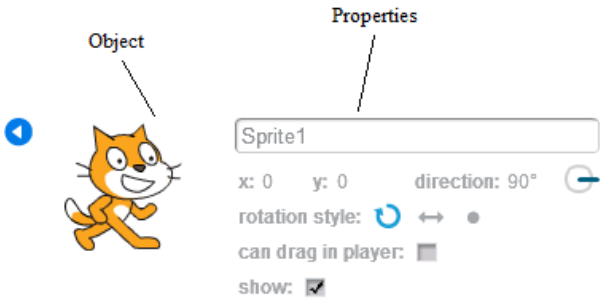

Figure 1. A Scratch object and its properties

Most of such properties can change their values by some method (script) or may be changed manually by a user.

Fig. 2 shows some blocks used to change properties of an object.

Figure 2. Blocks for changing the properties of an object

These blocks make it quite easy to understand the property concept.

B. *Building scripts with conditional statements and iterations*

Using Scratch blocks makes it easy to show students how cycles and if-operators work (Fig. 3).

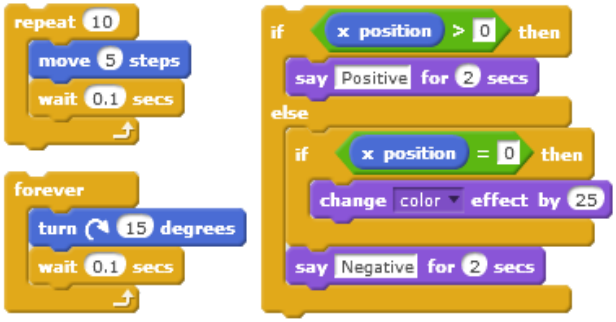

Figure 3. The cycles and branching realization

C. *Events*

There is a rather long list of pre-defined events that can be handled by Scratch scripts. An event is handled by a script (method) starting with a special block. Some of the event blocks are shown in Fig. 4.

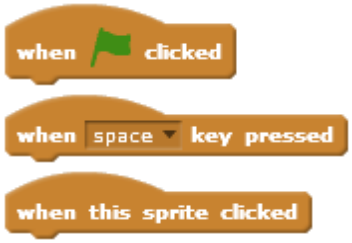

Figure 4. Some blocks to start the event handling script

D. *Parallel processing*

Running each script is considered a separate process. Scripts that handle the same event will be running in parallel after the event occurs. For example, scripts in Fig. 5 are performing in parallel and they implement the reaction on the "click" event. The first of them is responsible for moving the object, the second for changing its appearance.

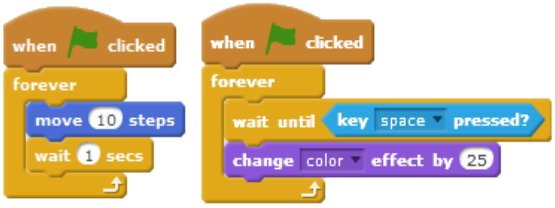

Figure 5. Realization of the parallel processes

Using Scratch scripts, students quickly obtain the concepts of parallel and coherent processing.

E. *Sending and receiving messages*

The message system provides communication and synchronization of the objects' behaviour.

Special command blocks broadcast a message from a script and any other script in the project can pick it up. Each object can send the message to other objects or to itself. The last instance is used to define the structure of an application.

There are two different blocks - one is for broadcasting and continuing, the other for broadcasting and waiting. The latter allows the process to continue only when all processes that picked the message have finished.

Scratch solves reactions to messages in the same way as handling other events. The script reacts to the message if it starts with a special block. (Fig. 6).

Figure 6. Blocks for sending and receiving messages

F. *Data*

Variable is one of the basic concepts in all programming languages. Its meaning in programming differs from its use in mathematics, which learners know from school. Scratch supports clarity in understanding the meaning of a variable concept as a named place in the computer memory. All variables have to be created manually before using them in a program code. Figure 7 shows the command "Make a Variable" in the Data group of the blocks.

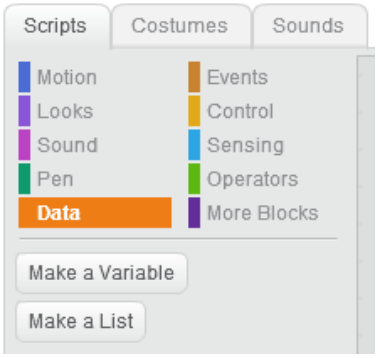

Figure 7. Commands of the Data group

The approach is very useful and worth considering in further programming activities.

Furthermore, the users have to define the scope of the created variable, which leads them to understanding the meanings of global and local variables and demonstrates the difference between these two (Fig. 8).

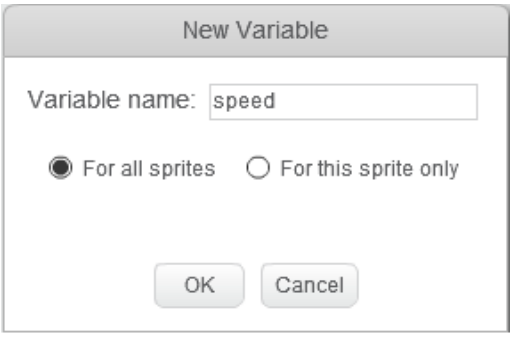

Figure 8. A variable declaration in Scratch

For example, if a sprite is active, only global and (its own) local variables can be used. Thereby students obtain the concept of the variable scope.

G. *Structured programming*

Structured programming is the most preferred approach in building programs. Students are encouraged to create a clear and structured program code. The ability to split a big task into smaller pieces plays an important role.

The majority of algorithmic languages support the definition of subroutines and functions, used in creating the code for the pieces of the project. One of the main methods of transferring data to subroutines is using the parameters. Experience shows that this is the most confusing topic for a beginner.

The new version of Scratch – Scratch 2.0 provides us with a perfect opportunity to make this issue easier. Learners can create and use their own Scratch blocks, where the definition of parameters is included (Fig. 9).

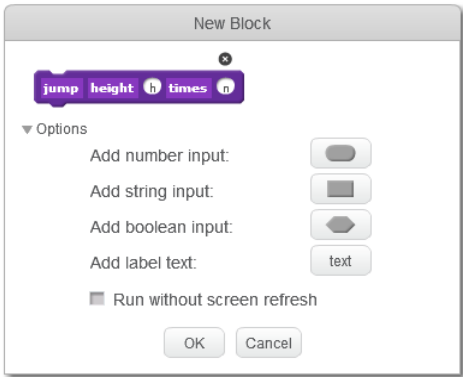

Figure 9. The user's block creation

Students learn to create a clear structure in their project. They divide their task into logical parts and create necessary user blocks, providing them with parameters. Now the main script can use standard and user-defined blocks, transferring the necessary data by means of parameters. Figure 10 shows the definition and calling of the user-defined blocks.

The issues reviewed here are very useful in the process of understanding modern concepts in building software applications and, hopefully, help students in their future study and professional work.

The next step is to proceed with more complicated tasks in other programming systems. Practicing with Scratch tools makes this function easier.

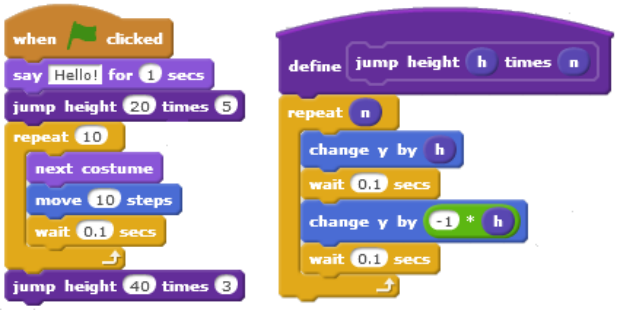

Figure 10. The main script and the user-defined block

V. SCRATCH + PYTHON

Python supports structured programming and procedural styles. In addition, Python does not require declaration of simple variables, which makes work with it easier for the beginners. It has a large and comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing the Python code to be executed on a majority of systems. It is an open source and is available to all students. The language is a high-level language and its syntax allows programmers to express concepts in fewer lines of code than would be possible in some other languages. There are a lot of tutorials and visualized debugging tools available. It makes it possible to provide learning support in different ways and everyone can find the most suitable one for themselves. On the other hand, this huge amount of information is often confusing and students need detailed guidelines from the instructor.

Python supports the object-oriented approach, but a lot of work can be done without it. It seems to be rather complicated for the beginner to orient in the documentation of the built-in classes and classes from different libraries. In our introductory course, we usually do not include the creation of classes and use only some of the existing objects. Therefore, the object-oriented approach is not taught and is limited to the possibilities provided by Scratch only.

VI. SCRATCH + VISUAL BASIC FOR APPLICATIONS

According to our approach, Visual Basic For Applications (VBA) in MS Office applications and Scratch have a lot in common. An Excel worksheet is a big canvas, onto which a number of graphical objects can be placed. There are a number of VBA methods and functions which have unambiguous equivalents in the form of Scratch blocks (e.g. set colour). We provide beginners with ready-made procedures, to be used as "black-boxes". For example, we have procedures, that correspond to Scratch blocks such as "wait", "move", "glide", "touching" etc. The primary control statements of VBA and Scratch are closely akin. All this enables a faster transition to more sophisticated tasks we undertake in VBA.

VBA, as well as Python, supports structured programming and provides the programmer with many built-in facilities. The text editing and debugging tools for the program code are visual and well observable.

However, using VBA cannot go far without introducing objects. There are many classes in the standard environment that can be used with their properties and methods.

VBA procedures are attached to the applications, so there are no interface problems (as the interface is included in the application itself). On the other hand, one has to use the object classes of the application for any input or output. However, some of the objects have a complicated structure and relationships with other classes, which is confusing for the majority of the beginners.

It was found out that using MS Excel for application creation is the easiest and most understandable way to practice the usage of classes. This is one of the reasons why MS Excel has been chosen as an environment for creating applications with VBA. The classes of worksheets and cells are comparatively easily applied. The expressions and many of the built-in functions in VBA have similar names and arguments with those in Excel. These topics are covered in the textbooks [11], [12].

VII. THE TEACHING METHODOLOGY

As mentioned above, we build UML (Unified Modelling Language) activity diagrams to describe algorithms in complicated tasks. We have been using a verbal description and pseudo code as well. Now, when we create Scratch projects as an introduction to programming, we can also use its scripts to visualize, formulate and describe the problem.

At the beginning, the teacher provides the students with a prepared model, which is analysed in a group. The analysis is followed by writing the program code according to the diagrams. Later on, students have to create the models themselves.

It has to be mentioned that according to tests [13] most of our students are visual learners [14], [15]. For them it is very important to see “how it works”. Scratch, with its elements of attractiveness, helps those students to understand the main idea of creating an application.

VBA already has a built-in visualising tool: students can follow the code execution using the Locals Window (Fig.11).

It automatically displays all the names of the declared variables in the current procedure, their types and their values. When the Locals Window is visible, it is automatically updated every time – students can see and check each step and its result in their program.

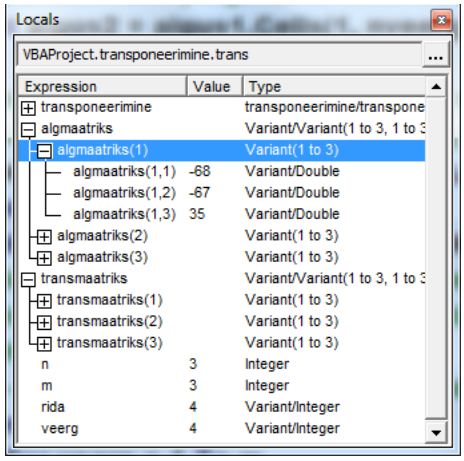

Figure 11. The Locals Window in VBA

Python does not have such an opportunity, but still needs to be visualised. We show students the Online Python Tutor [16]. Using the visualizing tool, the students can follow each step of their code and check the values and types of the variables, as well as the order of the operators during the execution. It has to be noted that this tool has some drawbacks, as it does not support the Python graphics, time functions and work with files. However, for the beginners in programming it gives the understanding of the code execution (Fig. 12)

Moreover, for our students we create short videos about the main terms, such as iterations, branching and the execution of the processes. It should be mentioned that we surely use sound and voice records in these videos to explain the complicated moments. In our work with educational visions we follow the ideas of Khan Academy [17].

In addition, we always try to provide students´ applications with a similar content. If a student has created a model in UML and the same application in Scratch, it is easier for him to "translate" it into the VBA or Python. Thus, if a learner understands the content of the model and the algorithm, it is easier also to understand the syntax of any language.

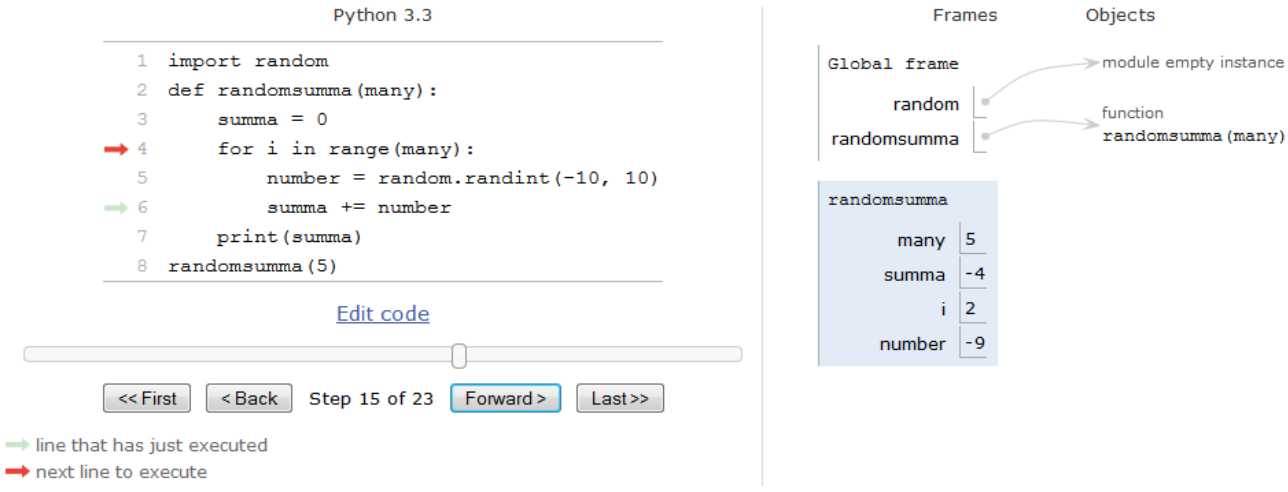

Figure 12. Python online visualizing tool

The following example is about solving a typical textbook task where the program generates a random number and asks the user to guess it. Here we see the steps of solving the task: UML activity diagram (Fig. 13), the script from Scratch project implementing the behaviour of the cat (Fig. 14) and finally the program code in Python (Fig. 15) and VBA (Fig. 16).

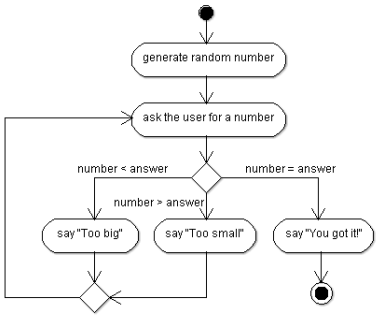

Figure. 13. UML Activity diagram

Figure. 14. Scratch project

```
from random import randint
number=randint(1,100)
while True:
    answer=int(input('Guess my number! '))
    if number==answer:
        break
    elif number>answer:
        print('Too small')
    else:
        print('Too big')
print('You got it!')
```

Figure. 15. Python code

```
Sub GuessTheNumber()
Dim answer As Integer
Randomize
Number = Int(100 * Rnd + 1)
Do
    answer = InputBox("Guess my number!")
    If Number > answer Then
        MsgBox "Too small"
    ElseIf Number < answer Then
        MsgBox "Too big"
    End If
Loop Until Number = answer
MsgBox "You got it!"
End Sub
```

Figure. 16. VBA code

VIII. CONCLUSIONS

Based on the above said, it should be concluded that in the Informatics course for the first year non-IT students we focus mostly on the model and algorithm, rather than teaching syntax and coding techniques. Visualized tools like Scratch are a good possibility to understand the main concepts of object-oriented approach and make it easier to write a programming code in any language when these concepts are clear. Similar ideas about Scratch and Python can be observed in [18].

In our course development, we try to keep up with the times and trends in computer education as [19], [20].

REFERENCES

[1] A. Robins, J. Rountree, & N. Rountree, 2003. Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2), pp. 137-172. http://dx.doi.org/10.1076/csed.13.2.137.14200

[2] A. Kak, 2014. Teaching Programming. Available at: https://engineering.purdue.edu/kak/TeachingProgramming.pdf

[3] CSTA K–12 Computer Science Standards. 2011. Available at: http://csta.acm.org/Curriculum/sub/K12Standards.html

[4] AP Computer Science Principles, 2011-2016. Available at: https://advancesinap.collegeboard.org/stem/computer-science-principles

[5] CSTA Computational Thinking Task Force. Available at: http://csta.acm.org/Curriculum/sub/CompThinking.html

[6] UK. The Royal Society. „ Shut down or restart? “ The way forward for computing in UK schools. Retrieved from: https://royalsociety.org/~/media/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf

[7] UK. Computing in the national curriculum: a guide for secondary teachers. Retrieved from: http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf

[8] MIT Media Lab, 2013. Scratch. Available at: http://scratch.mit.edu/

[9] Snap! Available at: https://snap.berkeley.edu/

[10] Blocky. Google Developers. Available at: https://developers.google.com/blockly/

[11] I. Amitan, J. Vilipõld, MS Excel rakenduste põhielemendid. Tallinn: Tallinna Tehnikaülikooli Kirjastus, 2000

[12] J. Vilipõld, MS Excel arendussüsteem Visual Basic. Tallinn: Tallinna Tehnikaülikooli Kirjastus, 2000

[13] B. A. Soloman, and R. M. Felder, (n. d.). Index of Learning Styles Questionnaire. Retrieved from: http://www.engr.ncsu.edu/learningstyles/ilsweb.html

[14] O. Mironova, T. Rüütmann, I. Amitan, J. Vilipõld, M. Saar, ”Computer Science E-Courses for Students with Different Learning Styles”, in: Annals of Computer Science and Information

Systems: Federated Conference on Computer Science and Information System, Kraków, 2013, pp. 735 - 738.

[15] O. Mironova, I. Amitan, J. Vendelin, M. Saar, T. Rüütmann, "Strategies for the Individualization of an Informatics Course", in: Annals of Computer Science and Information Systems: Federated Conference on Computer Science and Information Systems, Warsaw, 2014, pp. 835 - 840. http://dx.doi.org/10.15439/2014f259

[16] Ph. Guo. Online Python Tutor. Available at: http://www.pythontutor.com/

[17] Khan Academy. Available at: https://www.khanacademy.org

[18] C. Vorderman, Computer Coding for Kids - A Unique Step-by-step Visual Guide, From Binary Code to Building Games. London, DK Children, 2014.

[19] Exploring Computer Science. Retrieved from: http://www.exploringcs.org/

[20] National Science Foundation. Retrieved from: http://www.nsf.gov/

AUTHORS

O. Mironova is with the Tallinn University of Technology, Tallinn, Estonia as a lecturer (e-mail: olga.mironova@ttu.ee).

I. Amitan is with the Tallinn University of Technology, Tallinn, Estonia (e-mail: irina.amitan@ttu.ee).

J. Vendelin is with the Tallinn University of Technology, Tallinn, Estonia as a lecturer (e-mail: jelena.vendelin@ttu.ee).

J. Vilipõld is with the Tallinn University of Technology, Tallinn, Estonia as an associate professor emeritus (e-mail: juri.vilipold@ttu.ee).

M. Saar is with the Tallinn University of Technology, Tallinn, Estonia as an educational technologist (e-mail: merike.saar@ttu.ee).

Submitted, 19 May 2015. Published as resubmitted by the authors on 08 October 2015.