Programming Basics for Beginners

Experience of the Institute of Informatics at Tallinn University of Technology

https://doi.org/10.3991/ijep.v7i4.7425

Olga Mironova(✉), Irina Amitan and Jüri Vilipõld
Tallinn University of Technology
olga.mironova@ttu.ee

Abstract—The present paper demonstrates the teaching approach in programming basics course for novices: schoolchildren of different ages and schoolteachers. This programming course was developed at the Institute of Informatics at Tallinn University of Technology in Tallinn, Estonia and it based on many years experience in teaching programming for non-IT first year students. The main aim of the chosen teaching approach in the course is to raise the motivation and keep the learners' interest in programming field on the high level. The idea of developed teaching technique is the implementation of the visual programming before a serious textual coding. Furthermore, authors suggest readers some ways and methods to overcome learners' difficulties in the first stage in a textual coding.

Keywords—programming basics, beginners, Scratch, Python, Visual Basic for Applications

1 Introduction

In recent years has greatly increased the interest in Computer Science, particularly at secondary schools. In this regard, several countries have carried out thorough investigations of the use of information technology and courses on Computer Science in different educational institutions, which analyses have shown that most of the courses do not meet learners' 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 in USA [1]. It sets out the basic requirements for the various areas and levels of the CS 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" [2] created under the support of US National Science Foundation (NSF).

Documents mentioned above are based on the concept of "Computational Thinking" [3], [4], which defines general principles for describing the problems and solving them by means of software systems, including such concepts as abstraction and mod-

elling, 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" 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 United Kingdom schools curricula [7].

In this paper, presenting teaching approach to the programming course for novices, authors use the principles introduced above and tries to implement them in the best possible ways.

2 Basics of Applications Development and Programming

Computer Science is necessary at modern schools for nowadays children. Results of the research [8] provide strong evidence to justify learning Computer Science in middle schools. The improvements in learning, teaching efficiency, and affective factors are easily discerned in the transition to secondary school Computer Science.

Speaking about programming for beginners in Estonia, it should be noted that the non-mandatory course 'Basics of Applications Development and Programming' has been included in the curriculum of Estonian secondary schools starting from year 2011. The course was created based on the experience of Department of Informatics at Tallinn University of Technology in teaching students for many years and the approach takes into account their main challenges in the beginning of the programming study. In considered course development authors and educators try to keep up with the times and trends in computer education as [9], [10].

2.1 The course structure

'Basics of Applications Development and Programming' practical school course duration is 35 hours, basic tools there are Scratch [11], [12], Visual Basic for Applications (VBA) and Python [13]. The course consists of three components:

- **methods and tools for creating applications:** learners get the information about modern and in demand programming languages in the world.
- **modelling and algorithmic principles:** learners get theoretical knowledge about models and algorithms building and try to compose their first models in practice using UML (Unified Modelling Language) basics and Scratch.
- **programming bases:** learners get practical knowledge in more serious programming in Scratch and after there is textual coding using Python and/or VBA.

2.2 The learning process

The course starts with UML basics and creating applications in popular visual environment Scratch. Creating applications in this environment, pupils learn the programming foundations and get their own ideas for further programs. On this learning stage students often receive simple programs for further development. The whole class gets the same program or blank of the program with a certain scenario. The task of the students is to develop and complete the received application.

The next learning step is the textual coding using Python and/or VBA. The choice of the programming language depends on learners' needs, prior knowledge and skills. On the beginning of this stage pupils try to solve the same tasks as they were solved using Scratch. At this stage the syntax of the language is the most difficult subject for them. After, when it is clear, learners write their own programs or solve tasks that are more difficult.

2.3 The course directions

Thus, lecturers of the Department of Informatics at Tallinn University of Technology created and constantly develop the following programming courses for beginners:

- 'Programming basics' for children from 10 to 14 years.
- 'Basics of Applications Development and Programming' for school pupils from 15 to 18 years;
- 'Basics of Applications Development and Programming' for schoolteachers;

The first Programming Basics course is addressed to the younger generation with the aim to kindle their interest to IT and particularly to programming. Due to the age and level of knowledge learners of this course create applications only in visual environment Scratch. However, it should be noted that sometimes we meet primary school pupils, who already build interesting applications and games in Scratch and have very good knowledge in Python. For such pupils we try to compose special individual curriculum to keep their interest in study in this age group. Unfortunately, they cannot take part in a course for the next age group due to lack of necessary knowledge in mathematics. This special curriculum consists of the tasks, which can develop pupils' logical thinking and keep their motivation for the further learning.

The second course is aimed to learners, who are already interested in programming and want to enhance their knowledge. This course is dedicated to more serious mathematical tasks and complicated games. Often at this stage, students already have their own ideas and just need help in their implementation. This age group works mostly independently or in small groups. However, our main aim there is to order and systematize their knowledge for successful further teaching and learning.

The last one, for schoolteachers, is dedicated to didactic problems and their solutions. The main tools here are Scratch, VBA and Python. The volume of each of these components varies within 35 hours depending on the level of knowledge and skills, as well as the needs of learners. Scratch course for teachers and its outcomes are demon-

strated in [14]. Similar approach can be found in two-day workshop for teachers, which is described in article [15].

2.4 Learning and teaching materials

For named programming courses new learning materials [16], [17], [18] were specially developed at the Department of Informatics at Tallinn University of Technology. The short report about Scratch in TUT this paper reader can find in the presentation [19] from Scratch Conference 2013 [20].

For those learners who are interested in Scratch and want to develop their computational thinking, free/open-source web application Dr.Scratch [21] was designed in Spain [22]. Learners and instructors can use this tool to analyze Scratch projects and receive feedback on the quality of the programs [23].

Authors of this paper suggest to teachers, who are new in the teaching of Computing, to read the article [24]. This study is able to offer guidance to teachers on how develop their Computing teaching skills. In addition, schoolteachers can familiarize with paper aimed to propose e-Learning model as an educational concept to teach introduction to programming for secondary school learners [25]. The objective of this model is to determine the components involved by applying collaborative learning in e-Learning, to enhance students' motivation and understanding in the subject Information and Communication Technology, specifically in topic introduction to programming. In addition, authors could suggest to Scratch teachers ScratchEd community [26]. This is an online community where Scratch educators share stories, exchange resources, ask questions and find people.

2.5 The course promotion

Unfortunately, there are no enough teachers for teaching this course in Estonia. It is the biggest problem in named course promotion.

In order to use and develop this new course and after get good results in teaching pupils the first need is to teach schoolteachers. At the present moment the majority of Estonian schoolteachers do not use knowledge of programming or very often do not have it.

Furthermore, primary and secondary schoolteachers are also very interested in using Scratch for teaching pupils but they have lack of knowledge and experience.

People often do a mistake that such environments as Scratch is only for programmers, teachers of Computer Science or for interested children. It is one more reason to show to schoolteachers how they can use Scratch for all school subjects for making their teaching process more attractive and dynamic for modern children [27].

That is why now TUT Department of Informatics is teaching Scratch for schoolteachers. This course is much deeper than Scratch courses for students because it has more methodical and didactic problems and tasks related to different subjects. Educators are using different methods of teaching, like group works, lessons-presentations and making curricula for future work.

After the first part of the course, Scratch programming, those schoolteachers, who are interested in programming, can continue their learning with Python and/or VBA. Often they choose both programming languages with the aim to improve their skills and provide their pupils with quality knowledge.

3 Several tricks for motivation

It should be noted that some learners lose their interest and motivation after the first failure in textual coding. We often face this problem during the above-mentioned courses. Regardless of age, most learners are acutely experiencing mistakes and setbacks. The consequences of this is the lack of motivation for further studies. The course instructors' main aim there is always to be ready to help and try to avoid this situation.

To raise and keep students' motivation in the learning process it is necessary to make the routine learning process more attractive and dynamic for them [28], [29]. A research into the motivations of students for studying programming is described in the article [30]. Results raise a number of questions for the teaching of programming. Achievements of the study in learning motivation indicate that teachers need to provide opportunities for students to develop their thinking [31].

The best way to interest and involve students in the learning process - their active participation in it. It is necessary to mention here that active learning can support learners' development of the four capacities in many ways. For example, they can develop as:

- successful learners through using their imagination and creativity, tackling new experiences and learning from them, and developing important skills including literacy and numeracy through exploring and investigating while following their own interests
- confident individuals through succeeding in their activities, having the satisfaction of a task accomplished, learning about bouncing back from setbacks, and dealing safely with risk
- responsible citizens through encountering different ways of seeing the world, learning to share and give and take, learning to respect themselves and others, and taking part in making decisions
- effective contributors through interacting together in leading or supporting roles, tackling problems, extending communication skills, taking part in sustained talking and thinking, and respecting the opinions of others [32].

In the current section of the paper authors, relying on literature and personal experience suggest some ways for making programming more engaging with the aim of raising their interest.

3.1 Independent e-learning

All above-mentioned programming courses are realized in Moodle [33] e-environment. Authors have chosen Moodle for learning and teaching for three reasons:

- Moodle is increasingly used in schools and universities
- Moodle is used in Tallinn University of Technology
- Moodle continuously develops.

Learners work there independently. This independent learners' work mostly consists of grasping the new material, taking self-tests and other learning tasks.

Learning new material. For better understanding, the theoretical material should be presented to novices in simple and clear forms. A multitude of books and any other materials not mean that they are useful for students; moreover, this can be intimidating for the beginners.

During programming courses development and evolution authors have been trying to adapt theoretical teaching materials for the e-environment and deliver it to learners in most suitable forms. The main conclusions here are to provide students with small portions of the learning material and maximally visualize them. In this context, a small portion does not mean that learners will lack some knowledge – it means that they are provided with well-filtered materials, which are adapted to beginners. A great role here is played by short teaching videos that explain the main programming concepts and usage cases.

Self-examination of knowledge. After each new topic learners have to fulfil corresponding tests. Modern testing systems provide the course instructors with a variety of test types and, using them in the course authors have worked out a test system, which uses tasks similar to the pre-prepared program tasks.

As experience shows, the most effective type of the test, which greatly helps learners, is "fill in the gaps" type. Pupils get the problem description and the corresponding program text with some gaps and their task is to fill in gaps and check their results afterwards. Doing this, they learn the syntax and learn to understand the algorithm.

In addition, tests where students have to make the program text out of sentences and arrange them in the correct order, teach students to see and understand a model of the proposed tasks.

It is very useful for the beginners to check and correct their classmates' programs. This activity develops such an important skill as the ability to read and understand a code written by another person. Moodle environment provides teachers with this opportunity and they periodically use it in the course.

It should be mentioned that learners' independent work in Moodle is divided into stages, which are ordered and a transition to the next stage is possible only after finishing the previous one. Such course construction helps the course teachers to monitor the current situation in students' knowledge. Moreover, such system brings a competitive note into the learning process and makes it more attractive.

3.2 Contact lessons

Practice is one of the most important steps in learning the art of computer programming [34]. It should not be expected from the beginners that they immediately start to write a program code after the first explanations. The best option here is a simple copying task from the teacher's screen projected on the board. During this copying students do not think about why it is so, their interests are limited to the fact that they need to copy some text on time and afterwards check whether the program works. It is great if they are able to do it on their own, however, usually students are not able to check and correct it due to incomprehension of the solution. To make the situation more positive, authors has worked out some strategies that can help to involve students in the coding process during face-to-face lessons.

Start from Scratch. It should be mentioned that during programming courses, the majority of the created applications are small games that learners can play – one more motivating factor. Using Scratch, students make games and within the process, learn to understand their algorithms. Scratch gives an opportunity to test and, if necessary, correct the algorithm immediately without thinking about the syntax. Afterwards, when an algorithm is already clear, it is simple to translate it to VBA or Python, concurrently learning the syntax. Thus, a teaching tool like Scratch already adds an element of attractiveness to the course. Paper [35] suggests the effectiveness of similar approach to reduce initial difficulties for freshmen learning programming concepts.

Keep attention - learn through the game. As a group-work assignment during a face-to-face lesson it is possible to offer learners a possibility to play a game: they have the algorithm of a program and each student in the class should write one line in the code. The named method is quite controversial, but it is useful at the beginning of coding, when students just learn the basics. Afterwards, when each student has his/her own programming style, it is not so useful. However, it teaches to understand others' manners and proves and demonstrates why one solution can be better and more logical than another.

It should be noted that this is important knowledge in any subject, not only in programming. In addition, this game greatly helps teachers to maintain a high level of students' attention during the lesson.

Find and fix mistakes. The next assignment for students, which are successfully applied, especially before practical tests, is correction of mistakes in a program text. As usual, students have their task descriptions and a ready program, which does not work at all or does not work properly. The number of mistakes is also known. The mistakes are different: from simple misprints to syntax or logical errors. As the authors' experience shows, such assignments are useful if offered as pair work. Here the group work skills are developing among students and they learn to understand the program code. Besides, in such a way students learn the syntax by discussing it.

When new theoretical material is explained, the course teachers often use ready-made programs to introduce some topics to learners. In such a case, it is advisable to prepare, in advance, some mistakes in the code and within the discussion, correct them together with students. This way, new concepts are assimilated much better and

students learn to respond to different types of errors and afterwards are not afraid of them.

The teaching strategies mentioned above are the main types that authors apply in programming courses and they are aimed at raising students' interest in the programming subject by better engaging them into the learning process. As students' feedback shows, these methods work and bring positive results. It is necessary to apply different strategies in teaching, not only in programming, with the aim of varying students' learning and educators teaching experience and style.

4 Discussion

After TUT programming courses and applied techniques description it is necessary to give an overview of other programming courses, which use Scratch for teaching. This overview should help readers to recognize the differences between them and the course, which is described in this paper.

4.1 Scratch courses in the world

Currently a variety of the programming courses for novices is developed in the world. Some of them are free, some are not; some courses are online, some are using blended learning principles. They all are different, but programming using Scratch is their main principle. It is not possible to describe and analyze all the courses in the current research, however consider some of them.

Firstly, Harvard University new programming course CS50, which was launched in autumn 2015 [36]. This course considers C programming language foundations based on some Scratch blocks, which help to understand main programming principles. Scratch is used there during the first 2 weeks.

Secondly 'A Computer Science Principles Course' [37], developed by the UTeach Institute at The University of Texas at Austin. Summary table [38] shows that one of the programming environments of this course is Scratch.

In addition is worth paying attention that Snap! - Scratch language implementation is one of the programming environments in the course 'The Beauty and Joy of Computing' developed at the University of California, Berkeley [39].

Free online short course – 'Scratch - Teach Computer Programming in Schools' [40] introduces learners to Scratch and its advantages in an educational setting, reviews the features and functions of the Scratch interface, and how to program using the Scratch software application.

'Introduction to Programming with Scratch in Education' course [41] at the University of Northern Iowa takes learners through the ins and outs of programming with Scratch and prepare for introducing students to Scratch.

The course 'Code Yourself! An Introduction to Programming' teaches learners how to program in Scratch, an easy to use visual programming language. More importantly, it introduces learners to the fundamental principles of computing and it helps them think like a software engineer [42].

One more course, which could be mentioned is free online course from Harvey Mudd College – 'Programming in Scratch' [43]. It should be noted that separate Python course [44] could be found there as well.

More different programming courses this paper reader can find in: [45], [46], [47], [48].

4.2 Scratch courses in Estonia

In Estonia there are not so many programming courses for novices, which use Scratch for teaching.

The University of Tartu has developed Scratch teaching material for students [49]. This educational material consists of fourteen units, which cover the main Scratch topics. Short videos with explanations and practical assignments are used there.

In addition there is one more 'Scratch' course [50], which consists of four parts with short videos, text materials and practical assignments.

Short report, which includes information about programming courses in Estonia can be founded in [51].

4.3 Analysis

The above-mentioned review shows that programming courses can be divided into two types:

- courses, which include Scratch as the main programming tool
- courses, which include Scratch (more precisely, its elements) as the introductive tool to a serious textual coding.

The first type has been mainly developed to meet the interests of younger students. No previous programming skills are required there. In these courses, elementary school students are introduced to fundamental programming concepts. Students learn how to create animations, computer games, and interactive projects using Scratch. Throughout these courses students create their own computer games and share them with their instructors and classmates.

The number of courses of the second type is far less. Those courses are aimed to university students and include some Scratch blocks for visual explanation of the basics programming concepts.

The teaching approach to the programming course, which is described in current paper, is slightly different. Despite the fact that the course 'Basics of Applications Development and Programming' is also an introduction to programming the main aim there is to develop students' algorithmic thinking using certain programming tools.

The TUT programming course combines four programming environments: UML, Scratch, Python and VBA, which proportions vary depending on the age of pupils, their level of knowledge and skills. Moreover, this course is able to be of interest to learners of all ages from 10 years.

Practical assignments also vary depending on the audience, however the teaching algorithm is the same: students construct models of the tasks and try to solve them using aforementioned environments. This approach focuses mostly on the model, algorithm and their visualization, rather than teaching syntax and coding techniques. The basic principle here: if a student is able to build a holistic model and create an algorithm, then student is able to implement it on any certain language.

5 Acknowledgment

Authors of this paper would like to thank the anonymous reviewers for their constructive comments and suggestions.

6 References

[1] CSTA K–12 Computer Science Standards. 2011. Available at: http://csta.acm.org/Curriculum/sub/K12Standards.html
[2] AP Computer Science Principles, 2011-2016. Available at: https://advancesinap.collegeboard.org/stem/computer-science-principles
[3] Jeannette M. Wing, Computational Thinking, Communications of the ACM, vol. 49(3), 33-35, 2006 https://doi.org/10.1145/1118178.1118215
[4] Report of a Workshop on The Scope and Nature of Computational Thinking, USA National Academy of Sciences, 2010
[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. Available at: https://royalsociety.org/~/media/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf
[7] Computing: a curriculum for schools, Computing at School Working Group. Available at: http://www.computingatschool.org.uk, 2011
[8] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari, From Scratch to “Real” Programming. ACM Transactions on Computing Education, Vol. 14, No. 4, Article 25, 2015.
[9] Exploring Computer Science. Available at: http://www.exploringcs.org/
[10] National Science Foundation. Retrieved from: http://www.nsf.gov/
[11] MIT Media Lab, 2016. Scratch - Imagine, Program, Share. Available at: http://scratch.mit.edu/
[12] Maloney, J., et al. The scratch programming language and environment. Trans. Comput. Educ., 10(4):1–15, Nov. 2010. https://doi.org/10.1145/1868358.1868363
[13] Python. Available at: https://www.python.org/
[14] Nathan Bean, Joshua Weese, Russell Feldhausen, R. Scott Bell. Starting From Scratch. Developing a Pre-Service Teacher Training Program in Computational Thinking. 2015. http://doi.ieeecomputersociety.org/10.1109/FIE.2015.7344237
[15] Patricia Haden, Joy Gasson, Krissi Wood, Dale Parsons. Can You Learn To Teach Programming in Two Days? ACE ’16 Canberra, ACT Australia. https://doi.org/10.1145/2843043.2843063

[16] Vilipõld, J., Antoi, K., Amitan, I. Rakenduste loomine ja programmeerimise alused. Valikkursus gümnaasiumitele. Available at: http://rlpa.ttu.ee/RLPA_opik.pdf
[17] Rakenduste loomine Scratchiga. Available at: http://rlpa.ttu.ee/scratch/Rakenduste_loomine_Scratchiga.pdf
[18] Rakenduste loomine Scratchiga. Available at: http://rlpa.ttu.ee/scratch/Scratch_20/Scratch_20_P.html
[19] The Use of Scratch in Estonia. Available at: https://drive.google.com/drive/folders/0Bwxop-tjmpUXdXBjNGZueHdsUzg
[20] Scratch - Connecting Worlds. Available at: http://www.scratch2013bcn.org/
[21] Dr.Scratch. Available at: http://www.drscratch.org/
[22] J. Moreno-Leon and G. Robles. Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. In Scratch Conference 2015, Amsterdam, The Netherlands, August 12-15, 2015, Proceedings, 48-53, 2015.
[23] J. Moreno-Leon and G. Robles. Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. https://doi.org/10.1145/2818314.2818338
[24] Sue Sentance & Andrew Csizmadia. Computing in the curriculum: Challenges and strategies from a teacher's perspective. Educ Inf Technol (2016). https://doi.org/10.1007/s10639-016-9482-0
[25] Vitri Tundjungsari. E-Learning Model for Teaching Programming Language for Secondary School Students in Indonesia. 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV). 262-266 https://doi.org/10.1109/REV.2016.7444477
[26] ScratchEd. Available at: http://scratched.gse.harvard.edu/
[27] M. Resnick, J. Maloney, A. Monroy-Hernández, and others, Scratch: Programming for All, Communications of the ACM, vol. 52 (11), pp. 60-67, 2009 https://doi.org/10.1145/1592761.1592779
[28] George Lucas Educational Foundation, 2014. Coding in the Classroom. Available at: http://www.edutopia.org/topic/coding-classroom
[29] George Lucas Educational Foundation, 2014. Project-Based Learning. Available at: http://www.edutopia.org/project-based-learning
[30] Jenkins, T. (2001). The motivation of students of programming. In Proceedings of ITiCSE 2001: The 6th annual conference on innovation and technology in computer science education. 53–56. https://doi.org/10.1145/377435.377472
[31] Lawson, R. J. The Effect of Viva Assessment on Students' Approaches to Learning and Motivation. International Review of Social Sciences and Humanities. Vol. 2, No. 2 (2012), 120-132.
[32] About active learning. Available at: http://www.educationscotland.gov.uk/learningandteaching/approaches/activelearning/about/what.asp
[33] Moodle Trust, 2016. Open-source learning platform. Available at: https://moodle.org
[34] Brenda Cheanga, Andy Kurniaa, Andrew Limb, Wee-Chong Oonc. On automated grading of programming assignments in an academic institution. Computers & Education 41 (2003), 121–131. https://doi.org/10.1016/S0360-1315(03)00030-7
[35] Roberto A. Bittencourt, David Moises B. dos Santos, Carlos A. Rodrigues, Washington P. Batista, Henderson S. Chalegre. Learning Programming with Peer Support, Games, Challenges and Scratch. https://doi.org/10.1109/FIE.2015.7344222
[36] David J. Malan, SC50. Available at: https://cs50.harvard.edu/
[37] A Computer Science Principles Course. Available at: https://cs.uteach.utexas.edu/
[38] Available CS Courses. Available at: http://apcsprinciples.org/
[39] The Beauty and Joy of Computing. Available at: http://bjc.berkeley.edu/

[40] Scratch - Teach Computer Programming in Schools. Available at: https://alison.com/courses/Scratch-Teach-Programming-in-Schools
[41] Introduction to Programming with Scratch in Education. Available at: https://uni-cs4hs-scratch.appspot.com/preview
[42] Code Yourself! An Introduction to Programming. Available at: https://www.coursera.org/learn/intro-programming
[43] Programming in Scratch. Available at: https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-1
[44] CS For All: Introduction to Computer Science and Python Programming. Available at: https://www.edx.org/course/cs-all-introduction-computer-science-harveymuddx-cs005x-0
[45] Scratch Programming for Elementary School Students. Available at: http://cty.jhu.edu/ctyonline/courses/computer_science/scratch_programming_elementary.html
[46] Programming for Kids. Available at: https://www.udemy.com/programming-from-scratch/
[47] Adventures in Scratch Programming Course. Available at: https://www.idtech.com/kids/tech-camps/courses/adventures-in-programming-with-scratch/
[48] Scratch1: Getting started with Scratch. Available at: https://www.yorksj.ac.uk/ils/digitaltraining/bookable-courses/programming-courses/
[49] Scratchi materjalid. Available at: https://courses.cs.ut.ee/t/scratch
[50] 'Scratch'. Available at: http://scratchime.weebly.com/
[51] Work done in Estonia for increasing society's commitment in ICT. Available at: https://sisu.ut.ee/sites/default/files/ict/files/eno_tonisson.pdf

7 Authors

Olga Mironova is a lecturer at School of Information Technologies at Department of Software Science at Tallinn University of Technology. Head of Study Centre for IT Foundations. Akadeemia tee Street. 15A, Tallinn 12618, Estonia. E-mail: olga.mironova@ttu.ee.

Irina Amitan is a lecturer at School of Information Technologies at Department of Software Science at Tallinn University of Technology. Akadeemia tee Street. 15A, Tallinn 12618, Estonia. E-mail: irina.amitan@ttu.ee.

Jüri Vilipõld is an associate professor emeritus at School of Information Technologies at Department of Software Science at Tallinn University of Technology. Akadeemia tee Street. 15A, Tallinn 12618, Estonia. E-mail: juri.vilipold@ttu.ee.

This article is a revised version of a paper presented at the EDUCON2017 conference held in Athens, Greece, 25-28 April 2017. Article submitted 14 July 2017. Published as resubmitted by the authors 03 September 2017.