Informatics in Education, 2021, Vol. 20, No. 2, 255–276

DOI: 10.15388/infedu.2021.12

A Holistic Digital Game-Based Learning Approach to Out-of-School Primary Programming Education

Taras PANSKYI, Zdzisława ROWIŃSKA
Institute of Applied Computer Science, Lodz University of Technology, 90-537, Lodz, Poland
e-mail: tpanski@kis.p.lodz.pl, zrow@kis.p.lodz.pl

Received: May 2020

Abstract. This paper presents an innovative educational approach to organizing the out-of-school teaching of programming in middle childhood. The proposed DGBL model includes three distinct educational phases, i.e. learning visual programming, programming and robotics, and programming and electronics. The research was carried out during the school years of 2017–2019. The study sample consists of 329 primary school students from K4 to K10 from the Lodzkie Voivodeship in Poland. The results were obtained from anonymous questionnaires completed by course participants. The answers confirm that the proposed approach helps children to learn the main concepts of computational thinking and programming. The described approach reinforces the essential idea in children that programming, engineering, mathematics and technology are intertwined in the modern world. Moreover, the approach combines and balances practical, methodological and pedagogical issues and is suitably integrated with out-of-school programming education to facilitate the teaching and learning process.

Keywords: out-of-school education, visual programming, digital game-based learning.

1. Introduction

In the 21st century, the digital literacy is important for each person. Computer education should be ensured at all levels of the educations system in the form which will enable not only passive understanding but also conscious participation of students and their own influence on the development of the digital economy in general. In the world, the demand for computer technology specialist is growing constantly; informatics courses at universities are very popular at the moment. However, the level of candidates for such courses is very diversified; there is a large group of students who have not learnt programming, logical and abstract thinking or even algorithmics before. It is important that even young kids and teenagers, who are "digital natives" nowadays (Prensky, 2001), learn how to use information technology expertise in other school subjects as well as their later professional careers. That way, they will be active participants in the digital society and, at the same time, they will be able to feel secure and more comfortable through the under-

standing of and critical approach to certain issues in the quickly-developing infosphere. However, to reach that goal, it is necessary to organize the whole education system correctly so that to ensure not only proper conditions for teaching, including the curriculum and information technology rooms equipment, but also properly trained teachers.

In 2016/2017, in Poland the New Core Curriculum was introduced, which is aimed at propagation of informatics among children and teenagers as well as their development and logic thinking. According to the Core Curriculum, teaching informatics with technology requires that teachers acquire the skills of using technology as well as pedagogical skills in order to implement technology to meet desirable learning outcomes. However, considerable uncertainties remain regarding the impact of instructional methods on technological integration and learning outcomes in middle childhood (Panskyi *et al.*, 2019). Moreover, teachers are faced with the problem of how to use effective pedagogical and technological methods in the middle childhood education settings. The new Core Curriculum has become more demanding with the progressive changes of education policies and the government support for informatics as an academic goal in Poland. However, there is a big difference between the implementation of competences intended by the Core Curriculum and the actual instructional practices of teachers. Some informatics teachers who have a full access to information technology seem to lack teaching competencies other than the use of PowerPoint and Word to provide and edit content information or the use of the Internet to find information. So far, teachers have remained the weakest link in the Polish primary education; therefore, Poland still has to wait to see the results of the new curriculum. Nevertheless, there is a gap which is temporarily filled with various out-of-school courses, holiday workshops etc.

This paper presents a new digital game-based learning approach with a complete set of visual and text programming courses for children attending primary schools in the Lodzkie Voivodeship, conducted by academic teachers (tutors) of the Institute of Applied Computer Science at the Lodz University of Technology (Poland).

2. Educational Models and Frameworks Behind the Teaching of Programming

Innovative educational models and frameworks which try to minimize the obstacles and fit the requirements and expectations of the new generation of children should use well-known learning approaches and develop the required programming skills and knowledge. Educational theories are considered useful tools in developing learning objectives and assessing children's attainments, but they are not applicable in this area directly. According to (Looi *et al.*, 2014) programming requires not only children's understanding of the relevant theory but also their ability to apply it to solving real problems posed by relevant life challenges. Therefore, traditional theories should be reevaluated in terms of their contribution toward the accepted objectives of education and new theories must be designed with the intention of helping children to grasp the main central concepts of programming in an implicit, experimental, and entertaining way.

Different researchers use different approaches to teaching programming based on different pedagogical theories, didactic frameworks, models, or educational approaches.

Recent years have witnessed another trend in educational technologies, which, as many have argued, are ushering in a new teaching paradigm. This paradigm is based on the use of the gamification process. Since children are familiar with Xbox's, PlayStations, and other popular gaming platforms, the Game-Based Learning (GBL) model aims at using games for learning of (not only) programming. Supporters of the GBL method argue that children should be prepared to meet the demands of 21st century by being taught to be innovative, creative, and adaptable. Therefore, the GBL method has resulted in the appearance of several frameworks proposed for the design of educational games. The four-dimensional framework proposed by (Freitas and Jarvis, 2006) comprises four basic principles: Context, Representation, Learner, and Pedagogy. The conceptual framework proposed by (Yusoff *et al.*, 2009) aims at providing a reference guide for the design of educational games. The "Design, Play and Experience" framework (Tekinbas and Zimmerman, 2010) aims at designing serious games for learning. The empirical gaming model framework describes learning as a circular process that constructs cognitive schemas through activities within a game's world (Kiili, 2005). The multi-dimensional framework (Song and Zhang, 2008) has been proposed regarding the proper design of educational games. The Educational Games Design Model Framework introduced by (Ibrahim and Jaafar, 2009) includes design, pedagogy, and learning factors when designing educational games. The Massive Multiplayer Online Role-Playing Game framework presented by (Malliarakis *et al.*, 2012) aims at familiarizing secondary school children who are novices to computer programming with concepts such as variables, if-statements, loops etc., and engaging them in algorithmic logic. For children familiar with coding, (Skalka and Drlík, 2018) proposed microlearning-based teaching framework for improving achieved programming skills.

Promoting the use of information and computer technologies (ICT) in tandem with the revolution in digital technology helps children to investigate how digital games can promote learning (Lynch *et al.*, 2015). According to (Steinicke, 2018) "Digital game-based learning is the process of being taught and/or learning via digitally enriched play-/game-like activities or by playing/designing/creating/modifying digital games." The literature on game-based learning has explored teaching effectiveness and student outcomes related to game design and development (Carenys and Moya, 2016; Coleman and Money, 2019; Trajkovik *et al.*, 2018). Digital game-based technologies appear to deliver one the most effective ways to ensure creativity in digital game-based learning (DGBL) environments. The research conducted by (Ott and Pozzi, 2012) offers wide evidence that digital games as creative techniques assist digital learning creativity. Dalal *et al.* (2009) mentioned that computer game creation is an innovative pedagogical approach to teaching computational thinking. Kebritchi *et al.* (2010) found a significance in teaching computer game-based mathematics to students. Rodríguez Corral *et al.* (2014) propose a game-based approach to the teaching of object-oriented programming languages in high school education. Similarly, at the high school level, (Hainey *et al.*, 2011) assess the use of a game to teach requirements analysis in software engineering. Dickes and Farris (2019) try to integrate informatics with the DGBL approach as part of an ensemble of STEM (Science, Technology, and Mathematics Engineering) work in the elementary classroom. Furthermore, many researchers admit that the knowledge and programming

skills acquired through the DGBL method are retained longer than information from other learning methods. Moreover, using DGBL in the classroom helps to engage children by involving them directly in the learning process. Therefore, in this paper, the authors investigate the game design features that promote the involvement and learning in DGBL settings. The aim is to identify how creative programming and design of game-based activities may affect children's learning and involvement, to develop a set of general recommendations for the enhancement of the DGBL educational outcomes, and to create a holistic and systematic approach to teaching the programming concepts in middle childhood settings. Moreover, this study begins an interdisciplinary dialogue between researchers, teachers, and children to determine and realize the vast potential of using the proposed programming approach to support DGBL.

3. Educational Approach to Learning Programming

The primary aim of this study is to propose an innovative and new educational approach to organizing the out-of-school teaching of programming in middle childhood. This kind of research is situated in a wide scientific context of design, development, implementation, and evaluation of educational DGBL methods and tools as a part of out-of-school education. The proposed approach combines and balances practical, methodological and pedagogical issues and is suitably integrated to facilitate the teaching and learning process. Moreover, according to (Steinicke, 2018) the proposed approach covers two distinct learning scenarios. The first scenario is learning by designing, creating and modifying digital games, when children are entertained and gain learning experience through their work. The second scenario is learning framed by playful interaction with digital media and game components. The proposed educational approach includes three distinct phases (learning programming, programming and robotics, programming and electronics). These phases are the core part of the whole educational approach (Fig. 1); furthermore, they are closely interrelated and they interact with each other at the same time. Moreover, the proposed solution is an interdisciplinary approach, where the authors highlight the role of programming in other disciplines, i.e. electronics, automatics and robotics.

This approach reinforces the essential idea in children that programming, engineering and mathematics are intertwined in the modern world. The reinforcement is in the understanding that these subjects are often taught in isolation while the fact is that they are all intertwined. Programming and mathematics lead to technology development, which is then integrated with engineering to make it useful in our everyday life. Tutors ask participants to pick any item from their household and try to fit it in a single STEM subject only. They will find it impossible to do so, for everything is integrated with each other. Every programming course combines artistic inquiry with scientific research and technological practice to explore the social, cultural and ethical potentials of emerging ICT technologies and challenges facing the smooth implementation of those technologies. Teaching foundational programming concepts, along with robotics and electronics, makes it possible to introduce children to important ideas that are involved in the de-

sign of many everyday objects they interact with. The proposed solution demystifies the programming concepts, explains their intellectual underpinnings, and covers the crucial elements in DGBL. Moreover, the flavor of DGBL integrated into STEM education resonates in several of the qualification standards: analyst (analysis of ideas and experience, discovery of open problems and challenges), researcher (understanding of the operation, performance, interaction of systems), designer (development of alternative and innovative solutions within given limitations and constraints), computing engineer (simulation, problem-solving using the most appropriate software environment), effective communicator (both oral and written, in a native and foreign technical language), open-minded person (the ability to think outside the national frame, life-long education), and even more.

Fig. 1 demonstrates the main components used to formulate the proposed out-of-school educational approach for learning programming. The presented approach includes five basic courses: Scratch programming, Robotics, Arduino basics, Arduino continuation, and Arduino advanced. Each course includes comprehensive descriptive information about the total number of boy and girl participants, duration of the course edition, average group size, number of groups, and course foundation date. The detailed descriptive statistics of each course are presented in the next sections.

Regardless of their previous programming experience, all participants start with the Phase 1 Scratch course. A basic understanding of Scratch visual-programming language will help participants to move fast on the learning track. It is important to emphasize that the participants could navigate through the courses only along the paths (arrows)

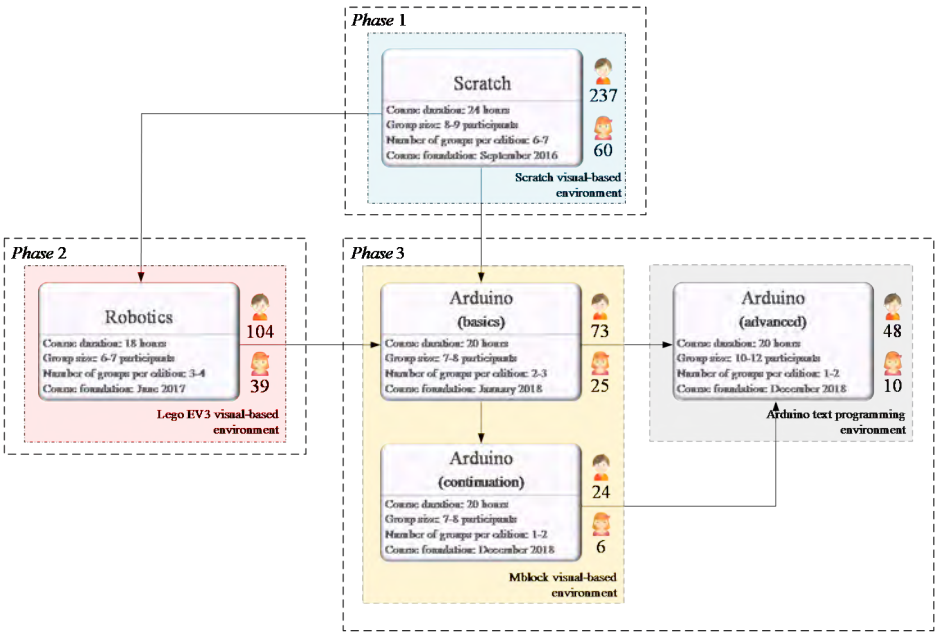

Fig. 1. DGBL approach to learning programming.

showed above. In other words, the participants could not shift between the courses in a random way. The aim of the paths is to soften the transition between the domain of block-based and text-based programming environments. Moreover, the paths aimed at preventing students' stress, fear, and emotional disorders related to a rapid change of programming languages. If the boundaries of the visual-based environment are quickly reached, the student could switch to a text-based language. However, each individual case is discussed in detail in tripartite order: parents, tutors and a participant. The shortest possible path between visual and text programming languages starts with Scratch, moves to the Arduino basics, and then follows to Arduino advanced courses. The longest educational programming path includes Scratch, Robotics, Arduino basics, Arduino continuation, and Arduino advanced courses. The participants could acquire new programming knowledge along individual learning paths that they have identified themselves.

Phase 1. Learning programming

There are many types of visual programming environments for beginner programmers that highlight distinct benefits of programming behaviors for problem-solving strategies in DGBL. Visual programming environments reduce the unnecessary syntax difficulties and assist programmers in visualizing the effects (Papadakis and Orfanakis, 2018). Such environments are able to enforce syntactic validity, which reduces error proneness in the notation by making it impossible for novices to encounter syntax errors (Chao, 2016).

The Scratch visual programming environment (Resnick *et al.*, 2009) has been chosen as a key DGBL software for teaching programming in Phase 1 of the proposed educational approach. Scratch is a free block-based learning environment tool designed for children aged 8 to 16 which aims at teaching programming principles such as loops, synchronization, variables, conditionals, operators, broadcasts, and more (Fields *et al.*, 2017; Stripeikaitė, 2017), through the use of graphical blocks which are overlapped by the "drag and drop" technique. Blocks are organized into different categories according to the functions they perform (motion, data, events, control, sensing, operators etc.). They are marked with different colors, which makes it easy to see the relationship between them. Moreover, the learning process can be integrated with geometric, arithmetic, and statistical concepts to investigate the key elements of solving problems and cultivate children's creative computational thinking for producing a joyful learning of programming.

The Scratch programming software is the most popular visual computing tool, more than doubling the next closest language (Blockly) according to (Rich *et al.*, 2018). However, there are several alternatives such as Kodu (MacLaurin, 2009), Lightbot 2.0 (Piteira and Haddad, 2011), and Alice 2 (Cooper *et al.*, 2003) for 3D modelling. More recent visual programming environments include Snap, Stencyl, Game Maker Studio and Tynker. Nevertheless, Scratch remains the most suitable visual programming language to develop computational thinking capabilities though programming (Maloney *et al.*, 2010). Moreover, according to (Kim and Ko, 2017; Sweigart, 2016), Scratch is the best and the most successful educational visual programming environment available today.

Phase 2. Programming and robotics

The second phase is related not only to teaching robotics but also to teaching through robotics. The use of robots in education goes beyond research to develop a new technology (Valls *et al.*, 2018) to integration of robotics in education and research in the field of teaching STEM. Moreover, teaching through robotics could be integrated as a complementary activity to the subject of ICT and in particular to the programming sessions. Additionally, the "jolly" aspect of programmable robots encourages children to be more creative by approaching the programming of a robot as a pleasurable leisure pursuit, significantly enhancing their willingness and determination to deal with programming (Rubenzer *et al.*, 2018). Phase 2 provides the opportunity to use children's previous programming knowledge in an unfamiliar task that requires them both to apply their acquired knowledge to a problem which is much more complex than their previous experiences and to introduce the additional element of mechanics, robotics, or mechatronics to it.

Currently, there are different kinds of educational robots on the market. They differ in terms of price, number of sensors, remote control capabilities, demands on the learner, appearance etc. The authors used Lego Mindstorms EV3 robotic kit for the Phase 2 course. EV3 robots can be programmed with a specific visual programming environment that is bundled with robot sets. Moreover, a Lego robot can be programmed using many other different programming languages or paradigms, such as C, C++, Java, or Python. Lego Mindstorms EV3 include Lego bricks that can control motors and sensors to perceive events or factors in the environment (temperature, distance, obstacles, light intensity etc.). The visual programming of an EV3 robot consists of dragging and dropping graphical colored blocks onto a programming canvas. Lego markets Mindstorms for children aged 10+ years. However, this downplays the enormous versatility EV3 has to offer to learners, professionals, and hobbyists of all ages (Vallance, 2016).

Despite popularity of Lego Mindstorms, there are already different kinds of educational robots, i.e. humanoid robots – the Aldebaran NAO robots (Gelin, 2017), Robovie (Ishiguro *et al.*, 2003) or Bioloid (Eaton, 2013), more simple Bee-Bot (Pekárová, 2008) and Dash and Dot (Eguchi, 2017), and more complex VEX robot (Tuluri, 2017). However, according to (Toivonen *et al.*, 2018) Lego Mindstorms EV3 is one of the most popular, moderately sophisticated, flexible in terms of functions, and sufficiently affordable educational robotics platforms with its unique comparative advantages, i.e. the start-up time for working with EV3 is very short; the assembly of the robot is very intuitive and no electrical wiring is necessary; there is a possibility to extend the provided components by integrating them with traditional Lego bricks.

Phase 3. Programming and electronics

The third phase is related to teaching electronics and uses children's previous programming knowledge and robotics experience. The Lego Mindstorms EV3 platform has its undeniable advantages; however, its disadvantages include its high price and limited

sensing capabilities. Therefore, the authors suggest the Arduino open source platform as the main hardware used in Phase 3. Arduino is essentially a computer built into a single chip and its brain is a microcontroller. There is a variety of Arduino boards available with different shapes, sizes and capabilities. Phase 3 is taught using the Arduino Uno board, which is officially endorsed by the Arduino team.

The Arduino basics and the Arduino continuation courses are carried out using the mBlock open source visual-based programming environment. The mBlock intuitive visual language is an adapted version similar to Scratch. Children can write programs by dragging and dropping building blocks in their already usual and habitual way. Furthermore, the mBlock interface allows one to control a variety of Arduino based programmable electronic projects. Children are able to design and program Arduino-compatible devices with multiple sensors, such as temperature and humidity, light, movement etc., to control LEDs and LED bars using dimmer switches, to output text information on an LCD display, to program a LED matrix, or to design and program a radar using a servo motor and an ultrasonic sensor.

The Arduino advanced course is a moment in children's education when they encounter programming tools resembling closer those used in technology professions. This course provides a jump from visual block-based programming to a text-based language. Children develop programs using the Arduino IDE text-based open-source software that can be programmed with C or C++ language. Children are able to repeat all the programs they design in Arduino basics and Arduino continuation courses described in an easy-to-decipher manner in a text-based programming language. Moreover, they learn the concept of arrays and recursion while adapting to new types of variables, conditional statements, and functions.

The main advantages of Arduino over Lego Mindstorms are the expendability, price, and sizes (Lazar, 2013). Apart from the price, what is crucial is sensing capabilities of the Arduino platform. EV3 can use the maximum of eight Lego sensors, while Arduino could easily rely on over a hundred sensors with many third-party and official built-in libraries. There are other small microcontrollers like Raspberry Pi, Udoo, or Beagleboard in the market line of devices (Donat, 2018). Each of these devices fills a particular niche and it can be difficult to compare them. However, Arduino is fantastic for creating simple projects and even controlling robots. Furthermore, Arduino could smooth the transition from block-based programming to text-based languages. Thus, the advantage of Arduino lies in its flexibility and endless possibility of its usage, enabling children to program it according their needs.

In recent years, there has been an increasing focus on the missing "S" of science, "T" of technology and "E" of engineering in middle childhood STEM curricula all around the world. Nevertheless, the current Polish Core Curriculum still neglects these crucial parts of STEM education. Instead, science curricula in primary schools are more likely to focus on the natural world including plants, animals, and weather, whilst the technology and engineering are forgotten altogether. According to the Sullivan and Bers (2016), "what is unique to our world today is the fusion of electronics with mechanical structures". The three-phase out-of-school DGBL approach offers a way to teach children about the types of electronics and mechanics they encounter in their daily life in

a hands-on and engaging way. In addition to teaching the concepts related to informatics, using robotics and electronics in out-of-school primary education can support the development of a range of cognitive and social milestones, such as cognitive motor and language skills, visual memory etc.

4. Course Organization and Participants

All courses were held in the Institute of Applied Computer Science at Lodz University of Technology. The Phase 1 Scratch course occurs cyclically twice a school year (autumn and spring edition). Each edition consists of 12 sessions, each taking 2 lesson hours (1.5 clock hour). All registered participants of each particular edition are divided into groups of 8–9 participants at the most. Course sessions are held only on weekends, that is every Saturday and Sunday. The comprehensive course organization is described in (Panskyi *et al.*, 2019).

The Phase 2 Robotics course occurs cyclically twice a year (winter and summer edition). Each edition consists of 6 sessions, each taking 3 lesson hours (2.15 clock hour). To avoid overcrowding, all registered participants of each particular edition are divided into groups of 6–7 participants per group. Moreover, trainers should devote more time not only to teaching programming but also to help children manually, as some children can learn and play without verbal or manual assistance, while others need major manual assistance to show a meaningful development activity involving some level of engineering design. Course sessions were held on working days, that is from Monday till Saturday. The sessions were held during one week around Christmas holidays in winter edition and August in summer edition.

The Phase 3 (Arduino basics, continuation and advanced) is held regularly twice a school year (autumn and spring edition). Each edition consists of 10 sessions, each taking 2 lesson hours (1.5 clock hour). In the Arduino basics and continuation courses, all registered participants are divided into groups of 7–8 at the most. In Arduino advanced, participants are divided into groups of 10–12.

The Phase 1 Scratch course is fully described in recent work (Panskyi *et al.*, 2019). The results were obtained from anonymous questionnaires completed by 221 course participants and their parents. Moreover, the quantitative analysis of students' Scratch finals projects has also been performed. The results show the importance of Phase 1 as an excellent tool for improvement of a child's computer competences, creativity, abstract thinking, or problem-solving strategies. Therefore, this work focuses on Phase 2 and Phase 3, while Phase 1 remains only to show the full picture.

From the international perspective, the study sample consisted of 329 primary school students from K4 to K10 curriculum standards in the Lodzkie Voivodeship (province) in central Poland. Since 2016/2017, the school system in Poland at the primary level consists of the 8-year primary school for students aged 6/7–15. Participants were recruited from public/private primary schools in the Lodzkie province.

Authors collected data for this sample for 3 years (see Fig. 1), i.e. during Phase 2 (from June 2017/2018 school year) – 143; during Phase 3 Arduino basics (from Janu-

ary 2017/2018 school year) – 98; during Phase 3 Arduino continuation (from December 2018/2019 school year) – 30, and during Phase 3 Arduino advanced (from December 2018/2019 school year) – 58.

With respect to gender, there were always more boy than girl participants, with 80 and 20 per cent respectively. In Phase 2 Robotics, 72.7 per cent of the participants were boys; in Phase 3 Arduino basics – 74.5 per cent; in Phase 3 Arduino continuation – 80 per cent; in Phase 3 Arduino advanced – 82.7 per cent. These courses are also aimed at collaborative work and team projects that tend to be gender neutral. Moreover, introducing DGBL in middle childhood may give girls a chance to become involved in engineering and STEM subjects before gender stereotypes have set at a later age.

In each phase, two tutors are always engaged in the teaching sessions and in directing the whole group of participants. The tutors work together and teach the same session topic at the same time. For example, one tutor presents a new topic and the other interjects with examples, explanations, and elaborations on key programming ideas. The tutors have the PhD degree in computer science and at least 5 years of pedagogical experience with regard to primary school education. Both of them can provide strategies to assist participants in remembering, understanding and organizing the presented information better. Moreover, tutors have sufficient pedagogical competencies in using different game-based approaches in programming courses.

5. Courses with DGBL Approach

The DGBL approach has certain undeniable advantages, as it is attractive for participants and enables playful learning of programming. Creating computer games also leads children to think about the complexity of their game design and implementation. The DGBL approach shows participants how to use different problem solving approaches and algorithms; as a result, they use various concepts of informatics while creating their games. The DGBL approach is manifested in each learning phase; furthermore, each of the above courses uses games as a tool for teaching programming.

The last session of the Scratch programming course (Phase 1) is dedicated to the final game presentation (Fig. 2 and Fig. 3). The final Scratch games are presented to the rest of the group and parents. The authors revealed that younger participants (9–12 years old) make maze-like, quiz-like and sports-like (football, swimming, skiing) games. Moreover, they choose realistic or fantasy adventure games. On the other hand, 12- to 14-year-old participants create more project genre variation (arcade games, fight simulations, tactical and popular on-time games like fortnight, counter strike, world of tanks etc.).

Throughout the Phase 1, the DGBL approach helps participants to learn how to use (create, change, delete) independent/dependent variables and lists in their games; helps to learn how to develop greater fluency with computational concepts (i.e. parallelism, events) and practices (i.e. iterative and incremental development); assists with becoming familiar with more complex Control (if-then, wait until, repeat until) and Broadcast block instructions; and helps to use More Blocks to reduce, organize and optimize their

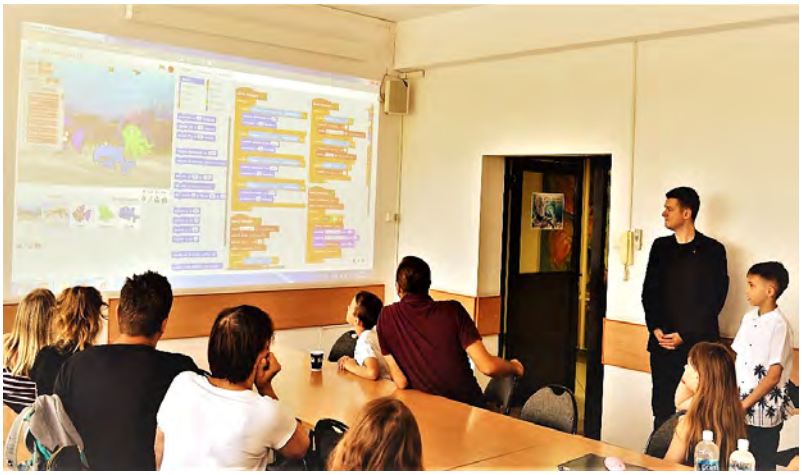

Fig. 2. The final Scratch game presentation by a 10-year-old participant.

Fig. 3. The final Scratch game presentation by a 14-year-old participant.

game space. Finally, by using the DGBL approach the participants learn how to implement algorithms for intelligent bots and dynamic rendered backgrounds in their computer pieces of art.

The Phase 2 of the robotics course starts with simple and creative tasks, such as making a robot spin around a table, driving a slalom between obstacles, or recognizing a given color, and more complicated games, i.e. a sweeper that cleans the table of Lego bricks, an EV3 puppy that guesses colors and barks, a maze runner and drag racing with ultrasonic sensor, line follower (see Fig. 4) etc.

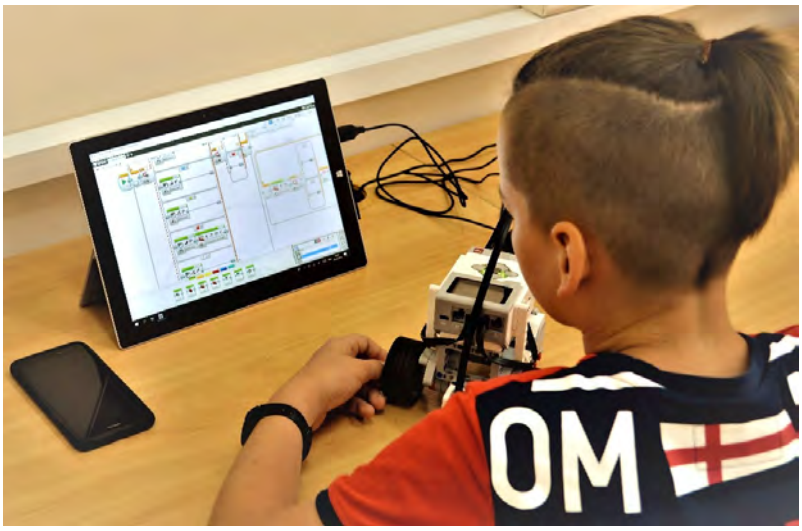

Fig. 4. Solving a line follower problem. Programming activity by an 11-year-old participant.

In the last session of the robotics course, participants are focused on learning programming using a space challenge contest. The space challenge playing field includes eight missions (*The Lego group*, 2020), i.e. communications station, satellite flight crew, crater and MSL, rock samples, solar panel, rocket and launcher, and mars outpost. All these missions (see Fig. 5) should be performed during the last session (2.15 clock hours). After the space challenge activity, each student chooses the achievement badge

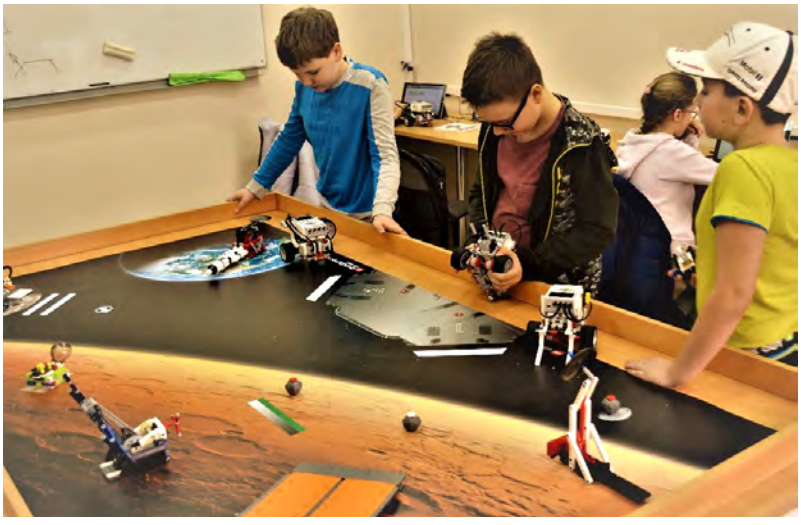

Fig. 5. The space challenge inspection activity performed by a group of 10/11-year-old participants.

which they feel represents their performance best, i.e. gold, silver, bronze, platinum, wooden medals etc.

The Phase 2 robotics course has a strong practical DGBL orientation and the participants are encouraged from the very beginning to feel free to propose solutions and alternatives in mechanical aspects, programming and optimization techniques, sensors to be used, time management etc. Furthermore, the DGBL approach fully meets the expectations in space challenge contest settings, where the participants of similar age and with similar interests cooperate using their creativity and problem solving techniques to solve different entertaining robo-gaming challenges.

As mentioned in (Dukish, 2018), a modern technologist should not only appreciate and understand general scientific concepts, but also be able to apply them to the everyday world. The Phase 3 programming courses with the DGBL approach can be thought of as a body of knowledge with electronics technology being the practical application of that knowledge. Arduino programming is introduced after a session dedicated to the study of some physical concepts, such as electrical current, resistance, voltage, and the relationship between them, i.e. the Ohm law. In the next sessions, students get a taste of the Arduino electronics; as a result, they are able to create simple projects using the Mblock environment, e.g. blinking LEDs, using a push button to switch LEDs, RGB LEDs, measuring ambient temperature and humidity, air pollution monitoring system etc. Using the Mblock environment with the DGBL approach, participants implement these simple projects and create more sophisticated games, such as traffic lights for sprites, maze-like games with a remotely controlled sprite (see Fig. 6), or an LCD screen game where a sprite jumps over hills and ducks under birds. During the Arduino continuation course, students increase their programming skills through the creation of advanced games, i.e. a classic snake game using a LED matrix and a joystick (see Fig. 7), a touch tic-tac-toe game, a tetris game using a LED matrix and a gyro sensor.

Fig. 6. A 10-year-old participant programming the maze. The creation process involves red barriers which will send a sprite to the beginning of the game-play if it comes in contact with them. The movement of the main character is implemented using a joystick.

Fig. 7. A 13-year-old participant programming the snake game. To make this game, a participant uses an 8×8 LED matrix for displaying the snake and its food and a joystick for moving in selected directions

The Arduino advanced course requires similar DGBL activities performed in the Arduino IDE text-based environment. The reproduction of visual blocks during the Arduino advanced course in a text form leaves out an important element of the experience of programming such as "seeing the same code from different perspectives" (Dickes and Farris, 2019).

Litts *et al.* (2017) and (Alimisis *et al.*, 2019) argued that most of today's educational robots are manufactured as black boxes "without understanding what's inside and how it works". These robots are appropriate for everyday gaming; however, they are ineffective in educational aspects where visibility and manageability promotes understanding of functionality and usability of computing. Unlike those robots, Arduino projects include elements of creative programming with decision-making and problem-solving strategies in conjunction with digital electronics components under a thick layer of DGBL activities and are, therefore, more accessible and informative for modern students.

6. Results

The data was collected using the quantitative method through web questionnaires. The questionnaire consists of 13 questions. The research was transversal, i.e. it covered a focus group at a particular moment. The first part of the questionnaire (Table 1) consisted of questions on the participant's programming learning outcomes (LO). In the second part the questions were grouped according to the tutor competencies (TC) towards teaching programming. The third part is dedicated to course organization issues (CO). The last question shows if participants want to receive notifications of new course editions for further personal programming development (FPD). The participants could

Table 1

Anonymous questionnaire provided to the participants

Question	Short name
Has the manner in which the class was taught helped you to develop your creative activity?	LO1
Has the manner in which the class was taught helped you to develop your skill of independent thinking?	LO2
Has the manner in which the class was taught helped you to develop your skill of algorithmic thinking?	LO3
Has the manner in which the class was taught helped you to improve your programming expertise?	LO4
Was the class taught in a comprehensible manner?	TC1
Was the support of the tutors sufficient?	TC2
Were the tutors committed to teaching the class?	TC3
Was the atmosphere during the class favourable for learning theory and skills?	TC4
Was the class held according to the schedule (e.g. punctuality, duration etc.)?	CO1
Were you treated seriously and with kindness during the class?	CO2
Have you received help from tutors in case of reporting any problems regarding the discussed topics?	CO3
Was it possible for you to make up for your absences on other dates specified by the tutors?	CO4
Do you want to take part in other classes in creative programming?	FPD

mark one of five options answering each particular question: definitely yes, rather yes, hard to say, rather no, definitely no. The questions were close-ended and accompanied with the Likert's scale, consisting of five-point rankings ranging from 'definitely no' to 'definitely yes'.

The results show that all participants had a significantly high mean score when answering all questions (see Table 2). Moreover, the mean score of each particular answer is 4.4–4.8, which indicates the range of rankings from "rather yes" to "definitely

Table 2

Descriptive statistics/questions

Short name	Robotics		Arduino basic		Arduino continuation		Arduino advanced	
	Mean	Std. Deviation	Mean	Std. Deviation	Mean	Std. Deviation	Mean	Std. Deviation
LC1	4.64	0.55	4.60	0.69	4.75	0.43	4.73	0.45
LC2	4.64	0.55	4.44	0.80	4.25	0.83	4.73	0.45
LC3	4.57	0.54	4.44	0.70	4.50	0.50	4.64	0.48
LC4	4.92	0.30	4.92	0.27	4.50	0.50	5.00	0.00
TC1	4.36	0.60	4.68	0.47	4.50	0.76	4.36	0.64
TC2	4.35	0.89	4.68	0.47	4.50	0.87	4.36	0.88
TC3	4.38	0.73	4.60	0.49	4.25	0.83	4.45	0.66
TC4	4.71	0.55	4.92	0.27	5.00	0.00	4.82	0.39
CO1	4.90	0.42	4.76	0.51	5.00	0.00	5.00	0.00
CO2	4.79	0.53	4.76	0.51	4.17	0.69	4.91	0.29
CO3	4.87	0.48	4.72	0.53	4.17	0.69	5.00	0.00
CO4	4.52	0.70	4.04	1.25	4.17	0.69	4.09	0.90
FPD	4.74	0.55	4.44	0.75	4.50	0.50	4.73	0.62

yes". Thus, summing up, the overall quality of the programming courses with the integrated DGBL approach remains high, regardless of the number of participants, seasons, or particular edition. Secondly, the participant's programming learning outcomes are closely related to teaching competences of tutors. Lastly, the presented phases help to ensure consistency and continuity from one course to the next. New courses also give tutors the opportunity to keep track with young people gifted in creative programming and problem-solving outcomes over the long term. Thus, the majority of high marks were given by the same children at different programming courses, which means that new courses or later phases fueled children's fascination and motivation, and met their expectation.

The technological competence is critical because modern technology creates great opportunities for teachers to motivate the students and inspire their curiosity, imagination, and interest. The competence means that teachers are aware of how to integrate educational technologies, provoke critical thinking, cultivate knowledge and creativity, and develop student understanding (Olstad and Rouhani, 2019). Technical skills and the ability to use specific programming tools are only two of many underlying competencies being part of the teacher's digital competence (Zhu *et al.*, 2013). However, the teacher's competence includes more than the technological competence, as teachers need to have basic programming competences and have to be able to make pedagogical and didactic judgments based on how technology can increase learning opportunities for students in programming courses. Therefore, the authors present the statistical evaluation in which we tested the correlation between the answers to the questions LO1–LO4 (programming learning outcomes) and the questions TC1–TC4 (tutor competences).

Table 3 shows that there are statistically significant correlations between the answers to the question LO1 and questions TC1–TC4. They refer to the children that were able to develop creative activity with the help of tutors. Moreover, Phase 2 and Phase 3 courses help children to increase motivation (Fowler and Cusack, 2011) and spark students' imagination (Tsur and Rusk, 2018). The strong correlation between the ability to think independently (LO2) and TC1–TC4 also speaks in favor of this claim. Creative programming can contribute to many psychological benefits, because it magnifies the child's creative ability and can enhance the capacity of independent thinking. The answers provided by the participants which were able to develop algorithmic thinking skills (LO3) is interesting and shows a strong correlation with their answers to questions TC1–TC4. This finding is in accordance with expectations because children grasped the concepts of algorithmic thinking such as abstraction through visualized procedural hands-on programming activities. The moderate correlation with the attitude expressed in TC1 Robotics and TC1 Arduino advanced can only be seen in the question LO4. In the first case, it is related to the very premise of the Phase 2 Robotics course, that is to say the main purpose of Phase 2 is to strengthen the programming knowledge acquired in Phase 1. The second case shows the smooth transition from the visual-based programming paradigm to the text-based one; once the children have crossed the chasm to text-based programming, it takes them forward on the road to progress and new challenges.

Table 3
Pearson coefficients of correlation between the participants' programming learning outcomes and tutors teaching competences

		LO1	LO2	LO3	LO4
Robotics	TC1	0.814	0.814	0.920	0.554
	TC2	0.950	0.950	0.991	0.780
	TC3	0.982	0.982	0.916	0.983
	TC4	0.988	0.988	0.931	0.977
Arduino basic	TC1	0.943	0.993	0.977	0.925
	TC2	0.874	0.969	0.972	0.840
	TC3	0.982	0.934	0.879	1.000
	TC4	0.943	0.995	0.997	0.901
Arduino continuation	TC1	0.957	0.946	0.722	0.722
	TC2	0.871	1.000	0.764	0.764
	TC3	0.943	0.802	0.612	0.612
	TC4	0.783	0.960	0.833	0.833
Arduino advanced	TC1	0.852	0.852	0.931	0.605
	TC2	0.955	0.955	0.986	0.792
	TC3	0.988	0.988	0.940	0.975
	TC4	0.988	0.988	0.940	0.975

Gender differences can also be observed in DGBL skills and programming abilities. The analysis revealed that boys are better in areas of math that involve logical and analytical thinking (math quizzes, geometry, coordinate system, algorithms). However, our DGBL approach has helped a significant number of girls to become equal or even superior in logical thinking. Moreover, our previous research (Panskyi *et al.*, 2019) dedicated to the Scratch programming course revealed that girls are better in data representation and data manipulation (more accurate representation of position, direction, size, color, numeric and non-numeric values etc.). Girls learn informatics through meticulous and systematic scrutiny of the epistemic material in a precise and detailed manner, while boys tend to create large and expansive games/projects without paying any additional attention to improving the visualization aspect. On the other hand, boys have a greater tendency toward high programming abstraction and design.

Some research (Kim *et al.*, 2014; Hatlevik *et al.*, 2015) has shown that girls perform better than boys in ICT-related assessments. However, according to (Litt, 2013) boys reported higher levels of ICT literacy than girls. Finally, (Hatlevik and Christophersen, 2013) research studies have not found gender differences in ICT literacy.

According to (Smith, 2010), boys are better in areas of math that involve logical and analytical thinking. However, (Lee *et al.*, 2017; Alkhadrawi, 2015) examined gender issues in computational thinking and creativity, and they found no significant gender difference. Moreover, some studies have reported that gender has no effect on programming education (Akinola 2015; Zhong *et al.*, 2016).

In Poland, the main reasons for the gender gap are still related to the lack of appropriate family and teachers' support, negative professional stereotypes and self-efficiency. Common professional stereotypes associate high-level intellectual ability (brilliance, genius, etc.) with boys more than girls. Bedyńska *et al.* (2019) showed the concept of intellectual helplessness of Polish girls. That concept applies the informational model of learned helplessness to difficulties in the acquisition of new and complex knowledge (algorithms, programming etc.).

While currently there are significant differences, with regard to gender, in the participation in informatics out-of-school activities, growing evidence suggests that a number of external factors influence girls' opportunities to participate in informatics activities and their continued involvement in fields such as programming, robotics, and automatics may be predicted. Efforts by educators, researchers and government organizations seem to have had a positive effect on making informatics attractive to girls.

6. Limitations and Future Research

This study has several limitations. In this research, the authors focused on the questionnaires that were used to provide a better understanding of participants' feedback instead of more intense forms of qualitative methodologies, such as face-to-face interviews. Moreover, the questionnaire does not include a crucial question about the flexibility of participants' knowledge and skills transfer between the domains of block-based and text-based programming environments. This question will help the authors get deeper insights into the inner feelings, self-awareness, and personal development of the course participants. Moreover, it may help tutors to soften the transition into text-based programming without encountering a gap in participants' learning.

Another limitation may be a relatively low number of participants, i.e. 329 primary school children, and a relatively low number of filled-in anonymous questionnaires i.e. 221. Yet, even though the authors should be cautious in the interpretation of DGBL findings due to a small research sample size, they provide us with strong confidence about the veracity of the discovered arguments.

Thirdly, all the participants come from the Lodz Voivodeship (province). Moreover, according to the participant geographical distribution, the majority of the sample population lived at a distance of no more than 20–30 km from the venue of programming courses (city of Lodz). Further research is required to show the situation for all Polish children to confirm that this study's proposed DGBL approach positively affects the implementation of the out-of-school informatics education policy in Poland.

And finally, the draft of this paper was finished in the winter of 2020. Within the following months, some important developments took place in Poland, due to the COVID-19 virus that was spreading throughout the world. Nevertheless, the authors adapted the out-of-school programming education to the online teaching process with the distance DGBL approach. Further research should be focused on the analysis of the different DGBL approaches, one used to teach face-to-face and the other with distance learning.

Conclusions

The DGBL provides a valuable context in which the strategic management of complex problems can foster creative thinking skills and show children how their decisions can have dynamic outcomes. Moreover, today the DGBL becomes an established standard tool for teaching and learning computational thinking and programming. This paper presents the holistic DGBL approach for learning programming in out-of-school education in primary school settings. Solutions using the DGBL approach may appear more an adjunct than a natural partner to classic teaching methods. However, we believe that when implemented effectively, the DGBL approach has the potential to advance students' problem-solving skills as well as logical and computational thinking significantly. Moreover, the integration of the DGBL approach into the school curriculum, teachers' continuous and relevant professional development, and students' engagement in application of programming activities and concepts in Polish primary schools will prepare all of them better for challenges of the 21st century. What was particularly interesting was the achievement of a balanced course curriculum that incorporates game designs, creative programming, and algorithmic as well as computational thinking skills.

The results of the questionnaires have led us to identify important issues with significant intersections between the tutors' programming competences and the children's learning outcomes. It is important that the students be encouraged to follow this DGBL educational pattern and create new games at different levels of programming expertise. In addition, students also perform non-programming tasks, as they create games and interact with tutors on on-programming issues.

Finally, our work also extends the argument that programming with the DGBL approach in Polish primary schools is an emerging interdisciplinary dialogue which unfolds within the production of knowledge in the modern ICT society.

References

Alimisis D., Alimisi R., Loukatos D., Zoulias E. (2019). Introducing Maker Movement in Educational Robotics: Beyond Prefabricated Robots and "Black Boxes". In: Daniela L. (eds) *Smart Learning with Educational Robotics*. Springer, Cham.

Alkhadrawi, A. (2015). *Gender differences in math and science choices and preferences*. Doctoral dissertation, The University of Toledo.

Akinola, S. O. (2015). Computer programming skill and gender difference: An empirical study. *American Journal of Scientific and Industrial Research*,7(1), 1–9.

Bedyńska, S., Krejtz, I., Sedek, G. (2019). Chronic stereotype threat and mathematical achievement in age cohorts of secondary school girls: mediational role of working memory, and intellectual helplessness. *Social Psychology of Education*, 22(2), 321–335.

Carenys, J., Moya, S. (2016). Digital game-based learning in accounting and business education. *Accounting Education*, 25(6), 598–651.

Chao, P. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95, 202–215.

Coleman, T., Money, A. (2019). Student-centred digital game–based learning: a conceptual framework and survey of the state of the art. *Higher Education*, 79(3), 415–457.

Cooper, S., Dann, W., Pausch, R. (2003). Teaching objects-first in introductory computer science. *ACM SIGCSE Bulletin*, 35(1), 191.

Dalal, P., Dalal, N., Kak, S. (2009). Learning Computer Programming with Game Design. *Proceedings of the First International Conference On Computer Supported Education*, 135–138.

Dickes A.C., Farris A.V. (2019). Beyond Isolated Competencies: Computational Literacy in an Elementary Science Classroom. In: Sengupta P., Shanahan MC., Kim B. (eds) *Critical, Transdisciplinary and Embodied Approaches in STEM Education*. Advances in STEM Education. Springer, Cham.

Donat, W. (2018). *Learn Raspberry Pi Programming with Python*, 2nd ed. Apress.

Dukish, B. (2018). *Coding the Arduino. Building Fun Programs, Games, and Electronic Projects*. Berkeley, Apress.

Eaton, M. (2013). An Approach to the Synthesis of Humanoid Robot Dance Using Non-Interactive Evolutionary Techniques. *2013 IEEE International Conference On Systems, Man, And Cybernetics*. 13–16 October, Manchester, UK.

Eguchi, A. (2017). Bringing Robotics in Classrooms. In: Khine M. (eds) *Robotics in STEM Education*. Springer, Cham.

Fields, D., Kafai, Y., Giang, M. (2017). Youth Computational Participation in the Wild. *ACM Transactions On Computing Education*, 17(3), 1–22.

Fowler, A., Cusack, B. (2011). Kodu game lab: Improving the motivation for learning programming concepts. *Proceedings of the 6Th International Conference On Foundations of Digital Games – FDG '11*.

Freitas, S.D., Jarvis, S. (2006). A Framework for Developing Serious Games to meet Learner Needs. *Proceedings of the Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*. 4–7 December, Orlando, Florida.

Gelin, R. (2017). NAO. In: Goswami A., Vadakkepat P. (eds) *Humanoid Robotics: A Reference*. Springer, Dordrecht.

Hainey, T., Connolly, T., Stansfield, M., Boyle, E. (2011). The differences in motivations of online game players and offline game players: A combined analysis of three studies at higher education level. *Computers & Education*, 57(4), 2197–2211.

Hatlevik, O., Christophersen K. (2013). Digital competence at the beginning of upper secondary school: Identifying factors explaining digital inclusion. *Computers & Education*, 63, 240–247.

Hatlevik, O., Guðmundsdóttir, G., Loi, M. (2015). Digital diversity among upper secondary students: A multilevel analysis of the relationship between cultural capital, self-efficacy, strategic use of information and digital competence. *Computers & Education*, 81, 345–353.

Ibrahim, R., Jaafar, A. (2009). Educational games (EG) design framework: Combination of game design, pedagogy and content modeling. *2009 International Conference On Electrical Engineering and Informatics*.

Ishiguro, H., Ono, T., Imai, M., Kanda, T. (2003). Development of an Interactive Humanoid Robot "Robovie" — An interdisciplinary approach. In: Jarvis R.A., Zelinsky A. (eds) *Robotics Research*. Springer Tracts in Advanced Robotics, 6. Springer, Berlin, Heidelberg.

Kebritchi, M., Hirumi, A., Bai, H. (2010). The effects of modern mathematics computer games on mathematics achievement and class motivation. *Computers & Education*, 55(2), 427–443.

Kiili, K. (2005). Digital game-based learning: Towards an experiential gaming model. *The Internet and Higher Education*, 8(1), 13–24.

Kim, H. Kil, H., Shin, A. (2014). An analysis of variables affecting the ICT literacy level of Korean elementary school students. *Computers & Education*, 77, 29–38.

Kim, A., Ko, A. (2017). A Pedagogical Analysis of Online Coding Tutorials. *Proceedings of the 2017 ACM SIGCSE Technical Symposium On Computer Science Education*.

Lazar, J. (2013). LEGO, Arduino, and The Ultimate Machine. In: *Arduino and LEGO Projects*. Berkeley, Apress.

Lee, J., Jung, Y., Park, H. (2017). Gender Differences in Computational Thinking, Creativity, and Academic Interest on Elementary SW Education. *Korean Association of Information Education*, 21(4), 381–391.

Litt, E. (2013). Measuring users' internet skills: A review of past assessments and a look toward the future. *New Media & Society*, 15(4), 612–630.

Litts, B., Kafai, Y., Lui, D., Walker, J., Widman, S. (2017). Stitching Codeable Circuits: High School Students' Learning About Circuitry and Coding with Electronic Textiles. *Journal of Science Education and Technology*, 26(5), 494–507.

Looi, H.C., Seyal, A.H., Darussalam, B. (2014). Problem-based Learning: An Analysis of its Application to the Teaching of Programming. *Proceedings of the International Proceedings of Economics Development and Research (IPEDR)*.

Lynch, R., Mallon, B., Connolly, C. (2015). The Pedagogical Application of Alternate Reality Games. *International Journal of Game-Based Learning*, 5(2), 18–38.

MacLaurin, M. (2009). Kodu: end-user programming and design for games. *Proceedings of the 4Th International Conference On Foundations of Digital Games – FDG '09*.

Malliarakis, C., Satratzemi, M., Xinogalos, S. (2012). Towards the Constructive Incorporation of Serious Games Within Object Oriented Programming. *Proceedings of the 6th European Conference on Games Based Learning (ECGBL 2012)*. 4–5 October, Cork, Ireland, 301–308.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions On Computing Education*, 10(4), 1–15.

Olstad, H., Rouhani, M. (2019). Reflection on How to Write the Learning Outcomes for an Online Programming Course for Teachers. *Lecture Notes In Computer Science*, 597–608.

Ott, M., Pozzi, F. (2012). Digital games as creativity enablers for children. *Behaviour & Information Technology*, 31(10), 1011–1019.

Panskyi, T., Rowinska, Z., Biedron, S. (2019). Out-of-school assistance in the teaching of visual creative programming in the game-based environment – Case study: Poland. *Thinking Skills and Creativity*, 34, 100593.

Papadakis, S., Orfanakis, V. (2018). Comparing novice programing environments for use in secondary education: App Inventor for Android vs. Alice. *International Journal of Technology Enhanced Learning*, 10(1/2), 44.

Pekárová, J. (2008). Using a Programmable Toy at Preschool Age: Why and How? *Proceedings of the international conference on simulation, modeling and programming for autonomous robots (SIMPAR 2008).* 3–4 November, Venice, Italy.

Piteira, M., Haddad, S. (2011). Innovate in your program computer class. *Proceedings of the 2011 Workshop On Open Source and Design of Communication – OSDOC '11*.

Prensky, M. (2001). Digital Natives, Digital Immigrants Part 1. *On The Horizon*, 9(5), 1–6.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. *et al.* (2009). Scratch: Programming for all. *Communications of The ACM*, 52(11), 60–67.

Rich, P., Browning, S., Perkins, M., Shoop, T., Yoshikawa, E., Belikov, O. (2018). Coding in K-8: International Trends in Teaching Elementary/Primary Computing. *Techtrends*, 63(3), 311–329.

Rodríguez Corral, J., Civit Balcells, A., Morgado Estévez, A., Jiménez Moreno, G., Ferreiro Ramos, M. (2014). A game-based approach to the teaching of object-oriented programming languages. *Computers & Education*, 73, 83–92.

Rubenzer S., Richter G., Hofmann A. (2018). Robotics Peer-to-Peer Teaching Summer School Project Involving University Students, Summer Interns and Middle School Students. In: Lepuschitz W., Merdan M., Koppensteiner G., Balogh R., Obdržálek D. (eds) *Robotics in Education*. RiE 2017. Advances in Intelligent Systems and Computing, 630. Springer, Cham.

Skalka J., Drlík, M. (2018). Conceptual Framework of Microlearning-Based Training Mobile Application for Improving Programming Skills. In: Auer M., Tsiatsos T. (eds) *Interactive Mobile Communication Technologies and Learning*. IMCL 2017. Advances in Intelligent Systems and Computing, 725. Springer, Cham.

Smith, I. (2010). *Boys, girls & learning pocketbook*. Teachers' Pocketbooks.

Song, M., Zhang, S. (2008). EFM: A Model for Educational Game Design. In: Pan Z., Zhang X., El Rhalibi A., Woo W., Li Y. (eds) *Technologies for E-Learning and Digital Entertainment*. Edutainment 2008. Lecture Notes in Computer Science, 5093. Springer, Berlin, Heidelberg.

Steinicke, M. (2018). Digital Game-Based Learning as Digitization of Learning Culture. In: Jat D., Sieck J., Muyingi HN., Winschiers-Theophilus H., Peters A., Nggada S. (eds) *Digitisation of Culture: Namibian and International Perspectives*. Springer, Singapore.

Stripeikaitė, I. (2017). "Skipping the Baby Steps": The Importance of Teaching Practical Programming Before Programming Theory. In: Alcañiz M., Göbel S., Ma M., Fradinho Oliveira M., Baalsrud Hauge J., Marsh T. (eds) *Serious Games*. JCSG 2017. Lecture Notes in Computer Science, 10622. Springer, Cham.

Sullivan, A., Bers, M.U. (2016). Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *Int J Technol Des Educ*, 26, 3–20.

Sweigart, A. (2016). *Scratch programming playground*. San Francisco. No Starch Press.

Tekinbas, K., Zimmerman, E. (2010). *Rules of play: Game Design Fundamentals*. Cambridge,The MIT Press.

The Lego Group (2020).
`https://education.lego.com/en-us/support/mindstorms-ev3/space-challenge`

Toivonen, T., Jormanainen, I., Tukiainen, M. (2018). An Open Robotics Environment Motivates Students to Learn the Key Concepts of Artificial Neural Networks and Reinforcement Learning. In: Lepuschitz W., Merdan M., Koppensteiner G., Balogh R., Obdržálek D. (eds) *Robotics in Education*. RiE 2017. Advances

in Intelligent Systems and Computing, 630. Springer, Cham.
Trajkovik, V., Malinovski, T., Vasileva-Stojanovska, T., Vasileva, M. (2018). Traditional games in elementary school: Relationships of student's personality traits, motivation and experience with learning outcomes. *PLOS ONE*, 13(8), e0202172.
Tsur, M., Rusk, N. (2018). Scratch Microworlds: Designing Project-Based Introductions to Coding. *Proceedings of the 49Th ACM Technical Symposium On Computer Science Education*.
Tuluri, F. (2017). STEM Education by Exploring Robotics. In: Khine M. (eds) *Robotics in STEM Education*. Springer, Cham.
Vallance, M. (2016). Establishing Meta-Learning Metrics When Programming Mindstorms EV3 Robots. In: Uden L., Liberona D., Feldmann B. (eds) *Learning Technology for Education in Cloud – The Changing Face of Education*. LTEC 2016. Communications in Computer and Information Science, 620. Springer, Cham.
Valls, A., Albó-Canals, J., Canaleta, X. (2018). Creativity and Contextualization Activities in Educational Robotics to Improve Engineering and Computational Thinking. In: Lepuschitz W., Merdan M., Koppensteiner G., Balogh R., Obdržálek D. (eds) *Robotics in Education*. RiE 2017. Advances in Intelligent Systems and Computing, vol 630. Springer, Cham.
Yusoff, A., Crowder, R., Gilbert, L., Wills, G. (2009). A Conceptual Framework for Serious Games. *2009 Ninth IEEE International Conference On Advanced Learning Technologies*.
Zhong, B., Wang, Q., Chen, J. (2016). The impact of social factors on pair programming in a primary school. *Computers in Human Behavior*, 64, 423–431.
Zhu, C., Wang, D., Cai, Y., Engels, N. (2013). What core competencies are related to teachers' innovative teaching? *Asia-Pacific Journal of Teacher Education*, 41(1), 9–27

T. Panskyi received his MS from the Lviv Polytechnic National University, Institute of Telecommunications, Radioelectronics and Electronic Engineering. Currently he is a PhD student at Lodz University of Technology, Institute of Applied Computer Science. His areas of interest are computational thinking, digital game-based learning, data clustering, creative programming, ICT.

Z. Rowińska received her PhD from Lodz University of Technology, Institute of Applied Computer Science. Currently she holds a position as an assistant professor at Lodz University of Technology, Institute of Applied Computer Science. Her areas of interest are digital game-based learning, cryptography, creative programming, design thinking, algorithmics.