Olympiads in Informatics, 2017, Vol. 11, 55–75

DOI: 10.15388/ioi.2017.05

Improving Teaching and Learning Computer Programming in Schools through Educational Software

Metodija JANCHESKI
University Ss. Cyril and Methodius
Faculty of Computer Science and Engineering
Rudzer Boshkovikj street, 16, 1000 Skopje, Macedonia
e-mail: metodija.jancheski@finki.ukim.mk, meto@ii.edu.mk

Abstract. Computer programming is the universal language of our planet and a basic literacy in the digital age. There is no doubt that learning computer programming at a young age is helpful for all students at least in their everyday life. The benefits of learning programming help young students to gain advantages in thinking, processing and communication. These benefits can support acquiring, developing and improving the 21st-century skills among youth.

One of the main challenges of scientists and educational practitioners in the field is how to make computer programming attractive and interesting for the students in primary and secondary schools. The use of various educational software could have positive impact on this issue.

There are many successful examples of educational software used in schools. This paper emphasizes the usefulness of Scratch, Logo, ToolKid and other similar education software tools in teaching and learning computer programming fundamentals. Some of the most important features of such tools, including immediate feedback (instant positive reinforcement), visually, block-based, text-based and object-oriented programming are explained in details.

The author presents several practical examples of how the educational software tools mentioned above can improve teaching and learning computer programming.

Keywords: computer programming, Scratch, Logo, ToolKid, computational thinking.

1. Introduction

The first steps in computer programming are very important. Older generations were taught the languages like Basic, Fortran, and Pascal and gradually move on to C language. But these programming languages require knowledge of well symbolic expression through mathematical and logical expressions. When children jump straight into a traditional programming language, they get bored and discouraged, because one of the biggest obstacles in learning a programming language is learning the syntax of the language (Vlieg, 2016). Therefore, programming has handled older students in primary

education. The modern computer languages, like Python, Delphi, C++, C# and Java, also require similar level of previous knowledge.

It was expected the appearance of new visual programming languages, including Visual Basic and Visual C to make easier the process of programming. Unfortunately, it turned out that they are not suitable for young programmers.

In today's digital world, coding is a fundamental skill alongside math and reading, but too few kids have the opportunity to learn to program because it is rarely taught in school (Tynker…, 2017). Today, at the time of the graphic interfaces and multimedia it is also hard for young programmer to work in text mode.

For a long time, the opinion prevailed that programming is necessary only for gifted math students and to those who will continue to deal with programming as their further profession. Fortunately, some leading scientists, inventors and educators recognize the benefits of programming for all students. They were aware that programming drives innovations, leading to success in other life areas. According Nicholas Negroponte from One Laptop Per Child Project (Sande, Sande, 2014):

> *"Computer programming is a powerful tool for children 'to learn learning.' ... Children who engage in programming transfer that kind of learning to other things".*

In the same spirit, Dan Shapiro, Robot Turtles inventor said (Gardner, 2014):

> *"Being able to program will make children better at whatever they do... No matter what you do, programming unlock doors for you, helps you express yourself, and helps you become more successful in anything you decide to do".*

Follows the chronological list of some notable programming languages with the year of development given in the brackets: Fortran (1957), Lisp (1958), Basic (1964), Logo (1968), Pascal (1970), C (1972), C++ (1980), Python (1991), Visual Basic (1991), Java (1995), JavaScript (1995), Delphi (1995), C# (2001), and Scratch (2003).

Before appearance of Scratch in 2003, as mentioned above, it was generally accepted that the first programming language for young programmers should be the language Logo. Because of its simplicity and features for quickly getting the effective graphical results, Logo gained great popularity in schools worldwide. Compared with Logo, the new programming language Scratch goes one step further in terms of simplicity, attractiveness and visual programming.

The question then arises as to what is the most appropriate age to start learn text-based programming languages, including Logo. The same applies to visual programming languages like Scratch. There is a continuous debate on these issues and we are witnessing various solutions worldwide. There are also many unanswered questions including whether students should learn to code in elementary schools and whether computer programming need be introduced to everyone in primary and secondary education or it should be optional.

First, there are examples where students aged eight learn text-based programming languages. Considering only the abovementioned fact that learning the syntax of the

language is one of the main obstacles, this approach demands high level of patience and adaptation of teachers.

On the other side, even the major universities use visual programming languages like Scratch in the introductory computer science subjects. According analysis done by Philip Guo in July 2014, University of California Berkeley, University of Wisconsin Madison and Brown University teach either CS_0 or CS_1 using the Scratch (Guo, 2014).

Finally, considering the importance of the programming as one of the most valued 21st century's skills, the main challenge is how to enable the kids to start programming before they can read. Generally accepted opinion is that programming is best learned early (Tynker…, 2017), which is also true for learning foreign languages. In other words, no age is too early to learn programming, or 'the earlier, the better'. It is clear that learning while sitting in front of a computer for a long time is not a proper solution for young students. Short interactive multimedia applications were proved as an alternative approach that offers easier entry into programming (Ghose, 2016). This approach includes visual programming languages and educational games.

Andrew J. Ko, a researcher at the Information School at the University of Washington believes that the age of ten is crucial for learning programming languages. He said (Ghose, 2016):

> *"Once kids are about 10 years old, they may be able to work with coding languages on a computer. Right around that age, children develop a more sophisticated theory of mind and are able to predict what others are thinking and feeling – which also means they are able to make models of what their snippets of code will produce".*

The issues related to select which programming languages to teach, when and in what order remains to be decided by the educational experts, scientists and practitioners, as well as, educational institutions, in accordance with their educational strategies and politics and the best practices worldwide.

The Table 1 is not finished. It should be completed with other "serious" programming languages for higher students' ages. According the author, the suggested next steps after the young programmers are mastering Scratch programming language is programming with Logo, Python, Perl or Delphi, to reach the final phase, i.e. programming in C++ and C#, as a programming languages for professionals. In this spite, it is useful to review other findings from the abovementioned data analysis. For each university, Philip Guo looked for CS_0 and CS_1 courses in the CS, CSE, or EECS department. Follows the results of his analysis: Python (27), Java (22), MATLAB (8), C (7), C++ (6), Scheme (5), Scratch (3). Obviously, Python was the most popular language (80% of the top 10 CS departments, and 69% of the top 39 CS departments) for teaching introductory computer science courses at top-ranked U.S. Computer Science departments in 2014. Note that Scratch was the only visual programming language that made this list.

Computer programming is a great intellectual hobby; it provides the same opportunity for creative, concrete work in mathematical thinking that drama or creative writing does for verbal thinking (Harvey, 1997). It is very important to ensure kids to have fun

Table 1

Programming languages with recommended age and short description

Programming language/game (recommended age)	Short description / Educational value
BeeBot (1+)	A simple, real-world toy that can teach kids the basics of coding. It uses simple left- and right-buttons on the robot, and kids should learn how to sequence their commands to get the BeeBot from one end of the room to the other, avoiding obstacles along the way (Ghose, 2016).
Robot Turtles (4+)	An actual, physical board game that surreptitiously teaches kids the basics of programming. The game teaches kids how to use directions to navigate their turtles through a maze to a tasty jewel (Ghose, 2016).
Light Bot (4–8)	An iPhone or Android app that teaches kids to navigate a robot through a maze, turning on lights.
Dash & Dot (5+)	A programmable robot pack that may be the best for slightly older kids, around age 8, who are already excited about programming (Ghose, 2016).
ScratchJr (5–7)	The free Android or iPhone app which allows kids to use simple icons to code their own interactive stories and games (Ghose, 2016).
The Foos (5–10)	The free iPhone application that uses simple icons with symbols, such as monsters, arrows and speech bubbles to solve adventures like chasing down a donkey thief or rescuing puppies lost in space. Kids can learn the basics in an hour (Ghose, 2016).
Tynker interactive courses (7+)	Tynker is an online platform that easily and successfully teaches students how to code through games and stories. Students learn the fundamentals of programming and design through Tynker's intuitive visual programming language (Tynker…, 2017).
Scratch (8 – 16)	Scratch is a simple coding language, completely free and open to use. It gets kids exposed to fundamental coding concepts, such as repeating loops and if-then statements using bright, color-blocked textual commands (Ghose, 2016).
Lego Mindstorms (10+)	Lego Mindstorm combines the LEGOs with motors, sensors and remote controls. With Mindstorms, young students can build robots that walk, talk and do as they command (Fichtner, 2014).

while they learn, so they stay engaged and continue learning and creating. Nowadays, there is no doubt that programming allows kids to be creative and helps the self-confidence building and developing among them. Programming improves mathematical skills in a fun way. It teaches problem-solving skills and helps kids visualize abstract concepts (Tynker…, 2017).

2. Scratch

2.1. *What is Scratch?*

Scratch is a block-based imperative, event-driven, dynamically-typed (whether data types agree is checked during program execution) and interpreted programming lan-

guage*. As a tool that offers "programming without proper programming", Scratch is simple and clear programming language. It is also object oriented, visual programming tool which does not require any prior programming knowledge and experience. Scratch provides a rich learning environment for people of all ages (Marji, 2014).

With this programming language young programmers can program (create) their own computer games, interactive stories, animations, simulations and other multimedia projects and then to share their creations online (Marji, 2014; Honey and Kanter, 2013; Pollock, 2014; Ford, 2009). Moreover, the Scratch website enable young people from around the world to learn from each other, to get and give feedback, to share interactive tutorials, guided tours, science experiments, book reports, online newsletters, and much more (Vlieg, 2016; Pollock, 2014).

This 14 years old programming language is available in more than 40 languages and used in more than 150 countries (Vlieg, 2016; Scratch…, 2017). More tinkerable, more meaningful and more social than other programming environments are the three core design principles established for Scratch (Vlieg, 2016). Creating with Scratch also encourages students to learn to think creatively, work collaboratively, and reason systematically – essential skills for success and happiness in today's world (Vlieg, 2016; Pollock, 2014). The main motto of the Scratch project is: Imagine, Program, Share.

Block programming with Scratch is relatively easy, even for young children, and it's a good way to enter the world of programming. With Scratch, young programmers easily understand the basic concepts of programming in a very effective and fun way. It has been used to introduce important computational concepts such as repeat loops, conditional statements, variables, lists, data types, events, and processes to students of many different ages, from elementary schools through universities (Vlieg, 2016). Scratch also enable students to learn important mathematical concepts and terms, including coordinates, variables, and random numbers. After learning Scratch, transition to traditional text-based languages can be done more easily (Vlieg, 2016).

According M. Resnick, one of the founders of Scratch software (Pollock, 2014),

> *"Scratch is more than a piece of software. It is part of a broader educational mission. We designed Scratch to help young people prepare for life in today's fast-changing society".*

Developed by the MIT Media Lab's Lifelong Kindergarten Group, Scratch was conceived as an educational language that would make programming fun and accessible to a new generation. The researchers at this Group believed that it was very important for all children, from all backgrounds, to grow up knowing how to design, create, and express themselves. Inspired by how kindergarteners learn through a process of experimenting, creating, designing, and exploring, the Lifelong Kindergarten Group extended this style of learning to programming in general and Scratch in particular. The primary goal of the Scratch initiative was not to prepare people for careers as professional programmers but to nurture a new generation of creative, systematic thinkers comfortable using programming to express their ideas. When you learn to code in Scratch, you learn important strategies for solving problems, designing projects, and communicating ideas (Vlieg, 2016).

---

* https://wiki.scratch.mit.edu/wiki/Scratch_Wiki_Home

2.2. *How Scratch Works?*

After launching Scratch, the students can start trying things right away. There is a default character (the Scratch cat), which already has some media to play with: two images that form a walking animation, and a "meow" sound. The students can start programming behaviors for the cat immediately: click the *move* block and the cat moves; click the *next costume* block and the cat animates; click the *play sound* block and the cat meows. The blocks start with reasonable default values for their inputs, so the user don't need to fill in any inputs (Honey and Kanter, 2013).

Scratch uses graphical blocks (puzzle-piece shapes) of code to represent programming commands. Instead of typing commands, a student can create Scratch program (project) by dragging, dropping and snapping graphical blocks of code into different sequences and combinations (stacks, scripts), much like snapping Lego bricks together (Vlieg, 2016; Honey and Kanter, 2013). The connectors on the blocks suggest how they should be put together (Vlieg, 2016). While pulling blocks from the palette, it is immediately obvious, from the shapes of the blocks, which blocks can relate to one another.

Unlike traditional text-based programming languages, there is no syntax to learn and the Scratch programmers are relieved from all worries about the syntax. Instead, the grammar is visual, indicated by the shapes of the blocks and connectors. Blocks snap together only if the combination makes sense (Honey and Kanter, 2013).

From a collection of simple programming blocks, combined with images and sounds, young students can create a wide variety of different types of projects. While creating Scratch projects, they typically engage in an extended tinkering process – creating programming scripts and costumes for each sprite, testing them out to see if they behave as expected, then revising and adapting them, repeatedly. Scratch has a range of features that allow programs monitoring as they run. Scratch scripts always highlight while they are running, so the students can see which code is being triggered when (Honey and Kanter, 2013).

2.3. *The Main Benefits and Advantages of Using Scratch*

The initial objective of the Scratch project was to encourage creativity, imagination and curiosity among the young students. The ultimate goal is to develop a shared community and culture around Scratch (Vlieg, 2016).

Scratch's visual programming environment enables students to explore areas of knowledge that would otherwise be inaccessible. It provides a full set of multimedia tools that can be used for creating wonderful applications, which can be done more easily than with other programming languages (Marji, 2014).

The greatest benefit of using Scratch is disclosure the interest in programming among young computer users. Today, understanding and mastery the complex machines

like personal computers requires several years of work and learning. It is therefore very important to provide an opportunity for learning programming in an easy and understandable way.

One of the things that are crucial to attract and excite young developers is the feeling of ruling with animation on the computer screen. Commands for graphics management in Scratch are extremely powerful. Scratch variables can be used to manage a variety of graphics and sound effects. Therefore, it is not surprising that many people recognize Scratch as a "tool to create simple multimedia". Although Scratch is a programming language for novices, it is highly oriented to graphics, sound effects and animations.

With Scratch and its code blocks, the students can control and mix graphics, animations, music, and sound to create interactive stories, games, simulations, art, and animations and even share their creations with others in the online community (Vlieg, 2016). One of the main advantages of Scratch is that operation of simpler program modules is much better understood graphically rather than textually.

Scratch is an excellent programming language for beginners as it allows creating very attractive programs even at the first acquaintance with the programming. It introduces a quite simple philosophy, thanks to which anyone can almost immediately create their first interactive game, a cute animation or some other vivid creation. Looking like with ease mastered the computer, the young programmer encourages himself for further work. We should put an emphasis on perfecting the developer technique, and on the control over the program, and not on the animation effects. The advantage of Scratch is that its programs are displayed graphically: the scripts are complex, consisted of colorful blocks. Thanks to this, it is easy to understand how and what they work while the syntax errors (misspelled commands) are disabled. The other thing that Scratch supports is the immediate feedback which is high valuable for young students.

In many ways, Scratch promotes problem-solving skills – important in all areas of life, not just programming. The environment provides immediate feedback, allowing you to check your logic quickly and easily. The visual structure makes it a simple matter to trace the flow of your programs and refine your way of thinking. In essence, Scratch makes the ideas of computer science accessible. It makes learning intrinsically motivating; fosters the pursuit of knowledge; and encourages hands-on, self-directed learning through exploration and discovery. The barriers to entry are very low, while the ceiling is limited only by your creativity and imagination (Marji, 2014).

2.4. *Disadvantages of Scratch*

The possibilities of Scratch for solving mathematical and logical problems are modest. Due to lack of commands, like Input and Read, Scratch will not provide more than an effective animation. Also Scratch will not teach about modern techniques of programming, so it is necessary to continue with learning other programming languages.

3. Logo

Logo is a complex text-based educational programming language. It was designed in 1967 as a tool for learning – about computer programming, but also about other domains – mathematics, language, art, music… (Logo…, 2017).

Many believe Logo language is a language of children. Strictly speaking, the Logo language is also suitable for children. Others feel that the Logo is used for drawing the figures, it is also true. But most important is that the Logo is a language suitable for solving wide range of tasks of programming (Дичева *et al.*, 1996).

Logo is a dialect of Lisp, the language used in the most advanced research projects in computer science, and especially for solving tasks in the field of artificial intelligence. It was developed by artificial intelligence researchers. Their idea was to see if they could use some of their experience with the problem of trying to get computers to think in order to help human beings learn to think more effectively – at least about certain kinds of problems. Since 1984, Logo is no longer the only member of the Lisp family available for home computers. Another dialect, Scheme, has become popular in education (Harvey, 1997).

In contrast to earlier programming languages, which emphasized arithmetic computation, Logo was designed to manipulate language – words and sentences. Like any programming language, Logo is a general-purpose tool that can be approached in many ways. Logo programming can be understood at different levels of sophistication (Harvey, 1997).

Initially, the software was used in grades 6 to 8 and was mostly valued for its animation and turtle graphics. Everyday work gave the teachers a deeper understanding of Logo and confidence in using the program. This change the initial notion of Logo as a "kid's language". Logo is increasingly used at the high school level and more and more teachers consider Logo a suitable instrument for their own work. Informatics teachers, sometimes with the help of students, create Logo-projects that serve as small-scale learning programs (Papert, 1999).

When computer classes were initiated in elementary schools, the teachers noticed that even young children can create complex and interesting programs despite the fact

Table 2

The most common Logo commands

Type of commands	The commands
Moving commands	FORWARD (FD), BACK (BK), RIGHT (RT), LEFT (LT), HIDETURTLE (HT), SHOWTURTLE (ST).
Navigation (position) commands	HOME, SETX, SETY, SETXY, SETPOS.
Color commands	SETPENCOLOR (SETPC), SETBACKGROUND (SETBG), SETFILLCOLOR (SETFC), SETFILLMODE, SETFILLPATTERN (SETFP), PENREVERSE (PX).
Drawing commands	CLEARSCREEN (CS), PENDOWN (PD), PENUP (PU), SETPENWIDTH (SETPW), SETPATTERN, PENERASE (PE), SHOW PEN, SHOW PENSTATE

that they have not yet learned what "structured programming" is. It has become widely accepted that Logo is the program of choice for introducing primary schools' students to computers (Papert, 1999).

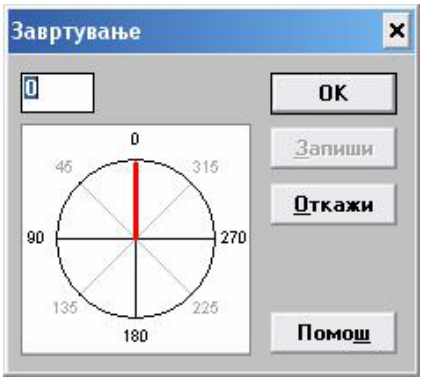
Logo should not be perceived as a mere technological tool, but as and innovative approach that encourages individuality and autonomy (Papert, 1999).

Logo language is widespread. It has some rich graphical capabilities. It's most known feature is the turtle (or small triangle) which movement can be easily managed (Дичева *et al.*, 1996).

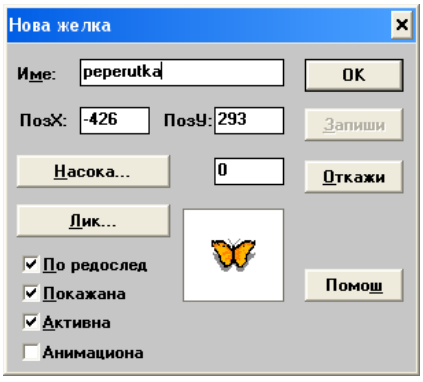
The Fig. 1 present some Logo windows.

The major role of Logo in schools, can be described as follows:

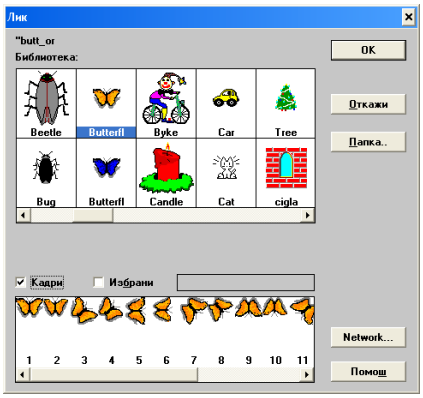
- Students learn Logo for learning's sake.
- Students study the basics of algorithms and programming. Logo is used as either the major environment for programming or as an example of a programming language. In elementary schools the goal might be phrased more modestly as "cognitive development".



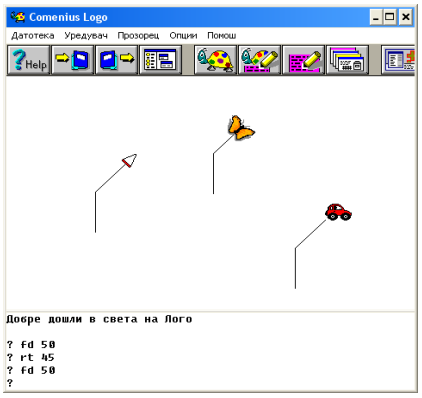
a) Selection of the angle of spinning.



b) Properties of the "new" turtle.



c) Selection of the favourite image.



d) Three active "turtles".

Fig. 1. Logo windows.

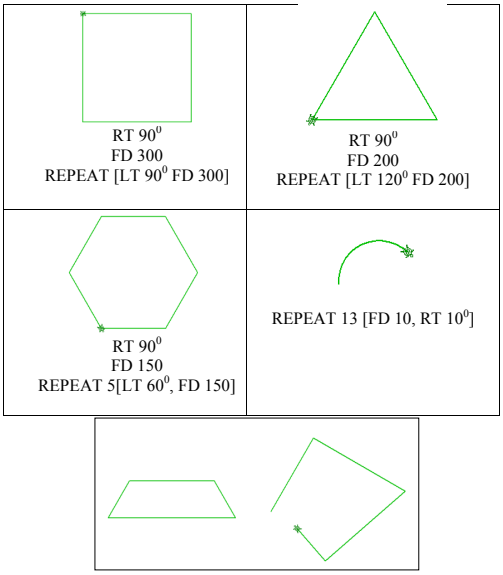


Fig. 2. Some geometric shapes in Logo.

- Logo is used to convince students that a computer is a convenient instrument for everyday work. Students create projects on elected themes and for other school subjects (Papert, 1999).

The Fig. 2 presents several geometric shapes drawn with the Logo tools, accompanied by a list of commands used for their drawing.

4. ToolKid

The ToolKid software, based on Commenius Logo is used in all primary schools of Macedonia (K4) since 2006. Macedonian version of this software was updated, expanded and adapted from Bulgarian version. Relevant curricula were developed and the software was installed in primary schools. Licenses for the introduction of this software were purchased by E-School Project. The Primary Education Project continue with ToolKid implementation and development where the E-School Project ended. Nowadays, the teachers are obliged to use ToolKid software during 30% of their regular classes.

This software did not pretend to change the existing tools, methods and techniques, recently used in the education, but its primary objective was to complement them, while enabling teachers and students to work in a new environment, providing plenty of opportunities, which offer new ways for answering the placed requests (Jancheski, 2016).

The ToolKID educational software could be useful in all subjects of K-4 education (mother tongue, mathematics, nature and society, society, art education and music education). Its organization is such that enables students to acquire solid informatics literacy and culture. The primary education is the right place where the students need

to obtain basic informatics knowledge. The curriculum needs to follow the informatics knowledge trends and to be redesigned in compliance with them (Jancheski, 2016).

The software package ToolKID contains 48 programs divided in 7 groups: Educational games, Drawing programs, Text programs, Sound programs, Animation and Video, Data Combining, Algorithms and Programming. These groups are presented in the main screen with images giving hints about the type of the programs included in them (Fig. 3).

The resources created with one program, like backgrounds, animation, images, sounds can be used in other in native and easy way by children. It is important and creative for them when they are going to work on different project topics.

The group Algorithms (Fig. 4) collects programs that develop the algorithm thinking (Pouring, At the river, Towers, Paths), mathematical knowledge (Figures, how many they are?) and lead to the world of the turtle geometry (Labyrinth).

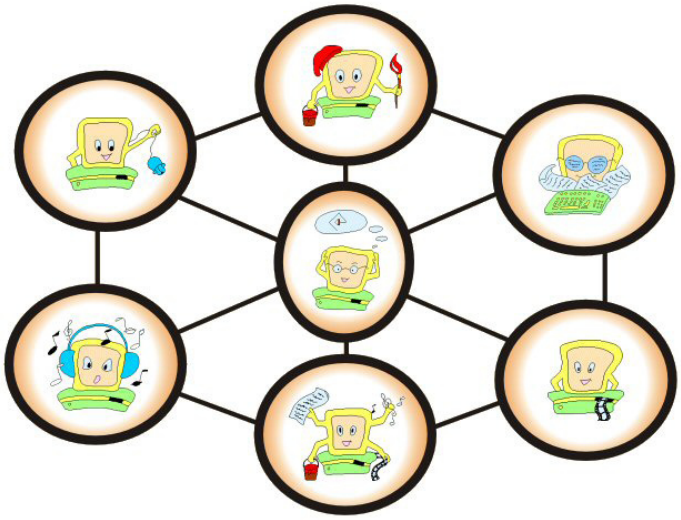


Fig. 3. Seven groups of Logo programs.

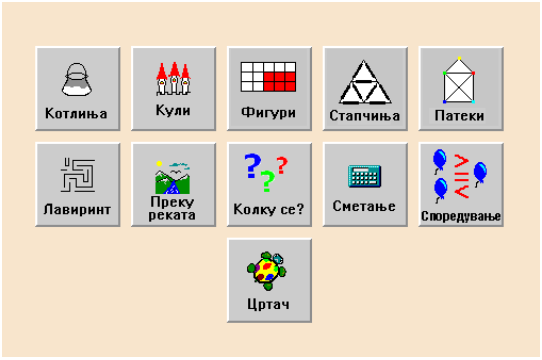


Fig. 4. Algorithms.

4.1. *Advanced ToolKid Usage*

ToolKid programs are designed for different class grades and they share common resources. The students can use certain ToolKid programs to create various multimedia files, which can be further shared by other ToolKID programs, or even to be used outside ToolKid environment. The modular structure of ToolKid will hopefully meet the requirements for usability at different age and different level of programming knowledge and skills.

The ToolKid software, especially the Teacher module, offers a variety of options for software development and new programs creation. By using Teacher Module, teachers can change various settings in each of the following programs: Silhouettes, Cat, Pie, Cards, Puzzles, Color, Painter, Music, Gossip, Clouds, Rebus, Notebook, and ABC. Teacher module allows teachers to share with their students these settings and other resources. This could contribute individual approach to different target groups or individuals and to improve active learning methods, including problem based learning, and project based learning. ToolKid also enables students to use the local PC resources, as well as, external multimedia files available on Internet (Jancheski, 2016).

4.2. *Contribution of ToolKid Software in Teaching and Learning Computer Programming*

Unfortunately, all of the above described programs, except ToolKid, are not a part of the regular curriculum in primary schools in the country. They can be used during extra work with gifted students who have an affinity for informatics and programming.

As stated above, TulKid programs are important in acquiring the contents of several school subjects. The main objectives of ToolKid software usage include:

- Acquiring the basic informatics literacy to the level of solving simple problems.
- Attaining a part of the curricula objectives for particular subjects by using information technology.
- Acquiring and developing logical and creative capabilities.
- Developing the regular attitude toward both computers and its programs using and protection.

ToolKid is not a typical software for teaching and learning programming, as Scratch and Logo, but it is very effective to be used as a preparation for programming.

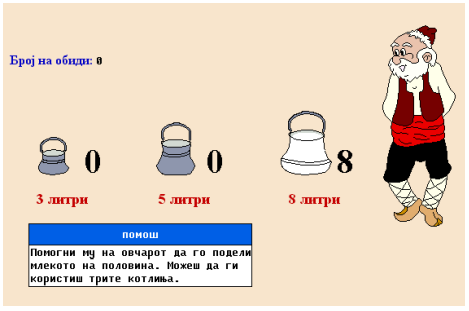
In order to program specific problems and tasks, students should be well knowing the nature and substance of these issues, including the entire range of possible solutions. ToolKid programs Pouring, At the river, Towers, and Paths allow students to study various problem situations and their variations through play and fun, in attractive visual environment. They are enabled to gradually reveal multiple sets of solutions and to explore for the best, quickest and most effective algorithms, programs and solutions. They should respect the rules and limitations of each of these programs. Horizons of

students should be expanded by considering variations of ToolKid programs involving more complex problem situations, rules, requirements and restrictions.

The majority of the K-4 teachers interviewed by the author during the phases of implementation and monitoring of ToolKid software were highly motivated to use this software and are eager for further professional development in the field. They usually are not programming professionals but they have to be aware about their role in development of algorithmic and computational thinking among students, which is crucial, according the author.

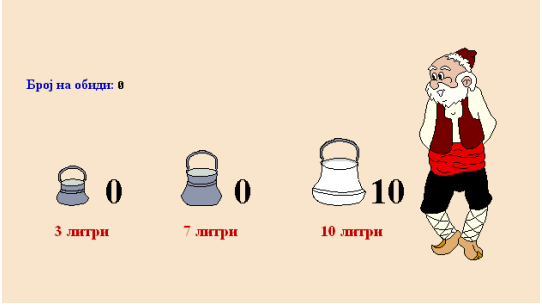
4.3. *The Pouring Game*

The shepherd has three bowls with the largest being full of milk. You need to help shepherd to share this milk fairly into two equal portions using only the three bowls and no other measuring devices. There are two levels of this game. The first one (Fig. 5 a) with three bowls with 8, 5, 3 liters, respectively, and the second (Fig. 5 c) with three bowls with 10, 7 and 3 liters, respectively. Each pouring from one to other bowl means one step. Determine a method of dividing with minimum steps.



a) Initial position (level 1).

b) Final position (level 1).



c) Initial position (level 2).

Fig. 5. The Pouring Game.

Table 3
One solution on the Pouring program (first level)

Bowl / Step	I 3 l.	II 5 l.	III 8 l.	Pouring
1	0	0	8	III → II
2	0	5	3	II → I
3	3	2	3	I → III
4	0	2	6	II → I
5	2	0	6	III → II
6	2	5	1	II → I
7	3	4	1	I → III
8	0	4	4	

The Table 3 presents schematic view of possible solutions on the first level of the program.

4.4. *Across the River (Game)*

The six tasks (Fig. 6) contained in this educational game belong to two variant of tasks.

Variant 1 (Fig. 7)

The shepherd and the characters (1. sheep, hay and wolf; 2. goat, cabbage and wolf; 3. rabbit, cabbage and fox) came to the bank of a river. The shepherd's challenge is to carry himself and all the characters to the far bank of the river.

The game requires the shepherd to get all the characters across a river in a small rowing boat. Only the shepherd can operate the boat. The boat can cross the river many times to get everyone across. As some of the abovementioned animals represent a real danger to other animals or objects, the sheep should be careful to do the task

Fig. 6. Menu with 6 tasks.

Fig. 7. Variant 1.

Table 4
Solution of Variant 1

N⁰ of trips	Moves	Left shore	Right shore
		S, R, C, F	
1	S, R →	C, F	S, R
2	← S	S, C, F	R
3	S, C →	F	S, C, R
4	← S, R	S, R, F	C
5	S, F →	R	S, F, C
6	← S	S, R	F, C
7	S, R →	/	S, R, F, C

Legend: S-Shepherd, R-Rabbit, C-Cabbage, F-Fox

keeping all characters intact. For example, if left unattended together, the wolf would eat the sheep and the goat, the fox would eat the rabbit, the sheep would eat the hay, or the goat would eat the cabbage. Solve this game in the smallest number of crossings.

Example with no solution (Fig. 8)**:** the girl and the characters: cat, mouse, and cheese.

Fig. 8. Example with no solution.

Variant 2

A grandmother and a grandfather of equal weight (90 kg), together with their two grandchildren, each of half their weight (45 kg), wish to cross a river in a boat that only accommodate a maximum weight of one adult (90 kg). If the total weight of passengers exceeds the capacity of the boat, it will sink. It is known that all four family members can operate the boat. What is the minimum number of trips all family members need to cross the river?

Subvariant: Without grandfather and with the same game conditions.

4.5. *The Tower (Mathematical Game)*

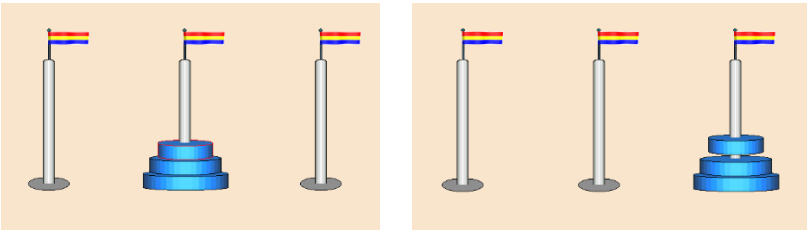
There are three vertical towers and n disks, ($3 \leq n \leq 7$). The initial position of the game is with all discs placed on one tower in ascending order of size. The goal is to move discs from this tower to another tower with minimal number of steps, while respecting the following rules: 1) each step consists of one move, 2) each move consists of taking the uppermost disk from one tower and placing in on the top of another tower, 3). No one disk may be placed on top of a smaller disk. The game provides counter of movements.

Before starting the game, the gamer has an opportunity to select the level, i.e. the total number of used disks. Allowed numbers are all elements of the set {3,4,5,6,7}.

When the gamer wants to move one of the disks, he/she need to click on it (after which its edges are stained in red), and then to click on the destination tower (where he/she want to place the disk). And without complying with the rule that it is not allowed to place greater on a smaller disk, the program will not allow it. In case the gamer wants to cancel the last selected disc before being moved, it can be done by clicking on the departure tower (the tower where this disk is placed).

The Fig. 9 provides a solution for n = 3 (three discs) where all three discs of middle tower should switch on the right tower with minimal moves.

We saw that with three disks, this mathematical game can be solved in seven moves. For a given number of disks, n, the minimum number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$.



a) Initial position. b) Final position.

Fig. 9. The Tower.

4.6. *The Paths (Mathematical Game)*

The program has three levels. Before choosing the level, select the desired figure out of four offered figures.

Level 1. Help the turtle to pass all sections of the path once by successively clicking on the colored circles. The turtle will automatically move after each subsequent click.

Level 2. Select the path that the turtle will pass by successively clicking on the colored circles. The turtle will carefully wait for you to press the Enter button, and then she will pass the path you have paved for her.

Level 3. Find out which circle should stand for the questionnaire. Then click on it and then press Enter. The turtle will start moving along the paved path, while leaving the painted trail down the path.

The first figure and the first level are predefined, as seen in the Fig. 10.

The Fig. 11 presents cases with different figures in three levels, respectively.

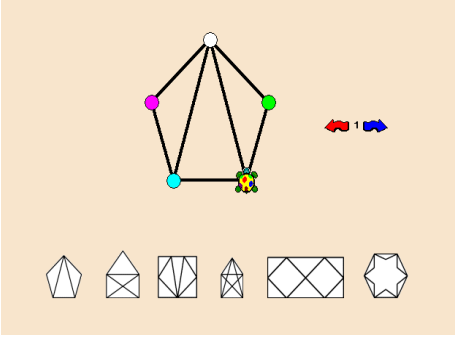
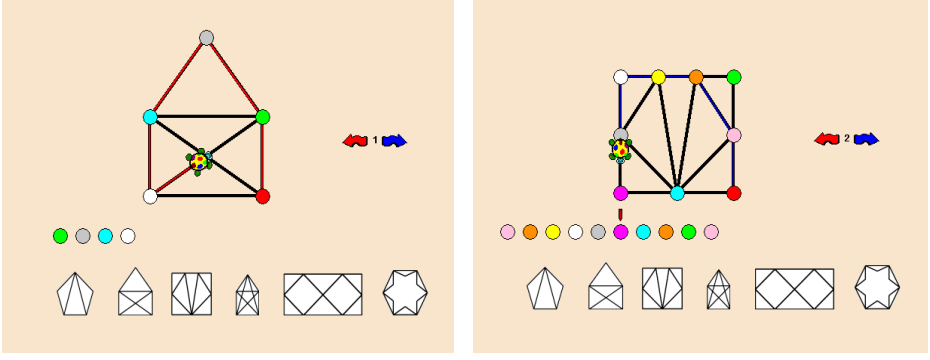


Fig. 10. Initial position of the game.



a) Level 1. b) Level 2.

Fig. 11. Cases with different figures.

4.7. *Labyrinth (Maze)*

Help the turtle to find her way through the maze from start to finish, while eating as much as possible food units.

The toolbar functions (Fig. 12) include:

- Moving one step forward/backward.
- Right angle rotations.
- Setting the speed of the movement (slow, normal, fast).
- Setting the step length.
- Selecting a maze from a set of mazes.
- Setting the number of fruits on the path.

Initially, the gamer determines the type of the maze, the step length and the speed of the turtle and the number of food items. The following figure shows a case with step length = 35, speed = 2, number of food items = 1.

Then the gamer uses the keys for navigation and movement to manage the turtle on the path to the goal. The turtle can never break down the walls which stand in the way in the form of black lines. If you direct to the wall, it will hit it and recover back one step.

The Fig. 13 presents a situation where the turtle successfully reached the target and delight with strawberries. The entire path the turtle passed from the start to the finish is colored in red.

Fig. 12. The toolbar functions.

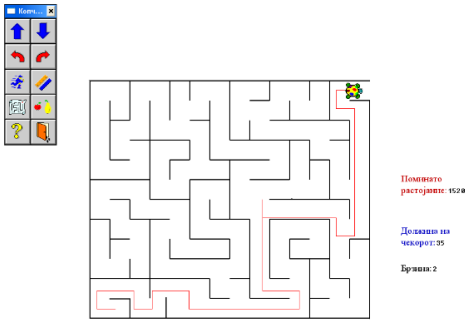


Fig. 13. Labyrinth 1.

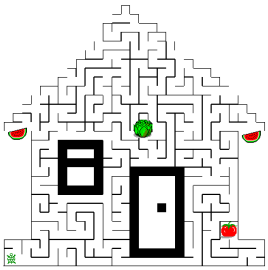


Fig. 20. Labyrinth 2.

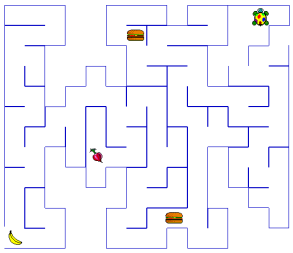


Fig. 21. Labyrinth 3.

Fig. 14. Examples with different initial parameters.

The Fig. 14 presents two examples with different initial parameters.

5. Conclusion

The development of the informatics industry, especially the computer technology, actualizes to a great extend the question for the promotion of the computer literacy as one of the crucial competencies of both the young people and the adults. This development every day increases the need of computer uses in the educational process and determine the criterion for literacy to be the mastery in working with computers. Enabling students for using the computer technology leads to approaching to the high educational standards which means contribution for the development of the brain freedom, knowledge, innovations and creativity as a base of the society.

Educational institutions, scientists, experts and teachers should be in line with actual achievements in science and new technologies. They also should be aware of and to promote 21st-century skills, including:

- Information and communication skills.
- Thinking and problem-solving skills.
- Communication and self-directed learning skills.
- Ability to use technology to access, manage, integrate, and evaluate information, construct new knowledge; and communicate with others effectively.
- Ability to learn academic content through real-world examples.

(Source: Partnership for 21st-Century Learning, 2004)

The programming skills are a part of skills listed above. They are widely recognized as a fundamental issue in today's digital word. Regardless of what programming language we use, it is important to know that programming allows us not only to do experiments with the computer, but also, by developing certain (good) style of programming, can help us to think better. Only it turns programming into activity worthy to be carried by people who do not intend to become specialists in informatics. The adoption of algorithmic way of thinking is an advantage in today's dynamic world, where people are forced to constantly plan (operations, finance, materials) and sometimes to operate in

conditions of chronic lack of resources of all kinds, including the information (Дичева *et al.*, 1996).

Scratch, Logo, and ToolKid are three successful projects in the field of computer programming for novices. The first two is directly related with programming, while the role of the third is latent; it could to be used as a preparation for programming.

Well-trained experienced teachers will know how to exploit the potential of such programs to promote algorithmic way of thinking as a part of computational thinking, and to encourage creativity and competitive spirit among the students. Critical thinking, problem solving and innovation are also issues where the role of teachers is unchangeable. The author fully agrees with the statement of Ed Lazowska, adjunct professor in University of Washington (Robot..., 2017):

> *"In the 21st century, computational thinking is essential for everyone. 'Computational thinking' is a problem analysis and decomposition, algorithmic thinking and expression, functions and abstraction, fault isolation and debugging".*

References

Дичева, Д.К., Дичева, Д.К., Николов, Р.В., Сендова, Е.Й. (1996). *Информатика в стил Лого*. Просвета.

Fichtner, A. (2014). *7 Interactive tools to teach your kids computer coding*. http://www.sheknows.com/parenting/articles/1048851/fun-ways-to-teach-your-kids-to-code

Ford Jr., J.L. (2009). *Scratch Programming for Teens*. Cengage Learning.

Gardner, A. (2014). *Teach Coding with Robot Turtles!* https://educators.brainpop.com/2014/05/23/teach-coding-robot-turtles/

Ghose, T. (2016). *The Best Coding Toys for Kids*. Live Science press. http://www.livescience.com/53957-best-coding-apps-and-toys.html

Guo, P. (2014). *Python is Now the Most Popular Introductory Teaching Language at Top U.S.* Universities. https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext

Harvey, B. (1997). *Computer science Logo style, Vol. 2. Advanced techniques*. The MIT Press.

Harvey, B. (1997). *Computer science Logo style, Vol. 1. Symbolic computing*. The MIT Press.

Honey, M., Kanter, D.E. (Eds.). (2013). *Design, make, play: Growing the next generation of STEM innovators*. Routledge.

Jancheski, M. (2016). One decade of ToolKid software implementation in Macedonian primary schools. In: *International Conference of Education, Research and Innovation Proceedings*. Seville, Spain, 2877–2886.

Logo foundation (2014–2017). http://el.media.mit.edu/logo-foundation/what_is_logo/index.html

Marji, M. (2014). *Learn to Program with Scratch: A Visual Introduction to Programming with Games, Art, Science, and Math*. No Starch Press.

Papert, S. (1999). Logo philosophy and implementation. Logo Computer Systems Inc.

Pollock, W. (2014). *Super Scratch Programming Adventure*. Canadá: The LEAD Project.

Robot Turtles (2014–2017). http://www.robotturtles.com

Sande, W. D., Sande, C. (2014). *Hello World! Computer Programming for Kids and Other Beginners*. Manning Publications.

Scratch Wiki (2017), https://wiki.scratch.mit.edu

Tynker – Learn to Code (2012–2017). https://www.tynker.com

Vlieg, E.A. (2016). *Lists. In Scratch by Example*. Apress, 223–248.

M. Jancheski, master of IT sciences, teaching/research assistant at the Faculty of Computer Science and Engineering, under the "Ss. Cyril and Methodius" University in Skopje. The leader of the Macedonian team on 7 Balkan Olympiad in Informatics and 5 International Olympiad in Informatics, since 1998. Consultant and trainer in the crucial national projects in the field of IT and informatics.