Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 106 (2013) 1576 – 1584

Procedia Social and Behavioral Sciences

4th International Conference on New Horizons in Education

Introducing basic programming concepts to elementary school children.

Goran Zaharija[a]*, Saša Mladenović[a], Ivica Boljat[a]

[a] *University of Split, Faculty of Science, Nikole Tesle 12, 21000 Split, Croatia*

Abstract

This paper discusses an approach to teaching programming that would allow elementary school children to adopt basic problem solving concepts. Adopting those concepts should help them in their further education, not only in programming but also in other areas that require logical thinking and problem solving.
Elementary school children are too young for traditional approach to teaching programming so in this paper we describe an approach appropriate for their age designed to avoid problems associated with young children such as short attention span. Preliminary experiment with 30 elementary school children was conducted and details and results are presented in this paper.

Keywords: Education, primary school, programming, robot, simulation, basic concepts…

1. INTRODUCTION

In the recent years, computers have become a standard part of majority of households (Seybert, 2012). This has resulted with new generation of children that are already familiar with computers at the beginning of their primary school education (Palfrey & Gasser, 2008). They possess elementary knowledge of computers and are already familiar with basic concepts. This fact allows for a different method of teaching those children, one that was not applicable to the past generations. Proper usage of computers as addition to the standard teaching methods can help enhance the learning experience for children by presenting them certain problems in a way that is appropriate for their age. Basic idea is to incorporate learning into something that children perceive as fun (Meluso, Zheng, Spires & Lester, 2012). By doing that, problems with lack of interest or misunderstanding of a

* Corresponding author. Tel.: +38521385286.
E-mail address: goran.zaharija@pmfst.hr

doi:10.1016/j.sbspro.2013.12.178

problem could be easily avoided. There are many different approaches for achieving that goal (Kirriemuir & McFarlane, 2004). This paper will present a brief review of some existing applications developed mainly for this purpose, and describe one approach that was designed specifically for primary school children. Purpose of that approach is to introduce basic concepts of programming and computer science to primary school children. There are two main reasons for selecting this kind of approach. First one is because recent studies have shown that children are able to learn while playing games (Prensky, 2008), both knowingly and unknowingly. The other reason is because children at this age, according to Piaget's theory of cognitive development (Wadsworth & Gray, 2004), are entering the concrete operational state of cognitive development. Concrete operational state is the third of four stages and it is characterized by development of logical thinking and problem solving skills which corresponds with skills and abilities we are trying to develop with described approach. To test this approach in real life conditions, preliminary experimental study was conducted in one primary school. Details and observations from that study are presented in this paper, as well as possibility for further experiments and improvement of that approach.

2. PROBLEM STATEMENT

Learning programming is considered hard. Programming courses in universities usually have highest dropout rates and are often regarded as most difficult (Yadin, 2011). This coincides with results available at our Faculty. There is still no clear understanding why some students learn to program easily and quickly while others have certain difficulties adopting this kind of knowledge (Wiedenbeck, LaBelle & Kain, 2004). In order to counter this, we propose that learning programming would be easier if students would already be familiar with basic programming concepts and it would be best if those basic concepts were taught in early age thus allowing students enough time to develop problem solving skills and logical thinking. As mentioned before, according to Piaget's theory of cognitive development primary school children are entering the phase in which they are beginning to develop logical thinking and problem solving skills. One of the difficulties regarding that development is that children at that age haven’t yet developed abstract, hypothetical thinking. Because of that, they can only solve problems associated with the real world or something they can relate to. Programming implicitly requires high level of abstraction and thus is hard to introduce to primary school children. To counter that problem, it is necessary to develop an approach to teaching that would make programming more accessible to the primary school children and would encourage the development of their logical thinking and problem solving, but at the same time would still be appropriate for their age.

3. STATE OF THE ART

There are already some applications available that are intended for teaching elementary programming (Robins, Rountree, & Rountree, 2003). In this chapter, short overview of some of those applications and their advantages and shortcomings will be given and they will be compared to approach proposed in this paper.

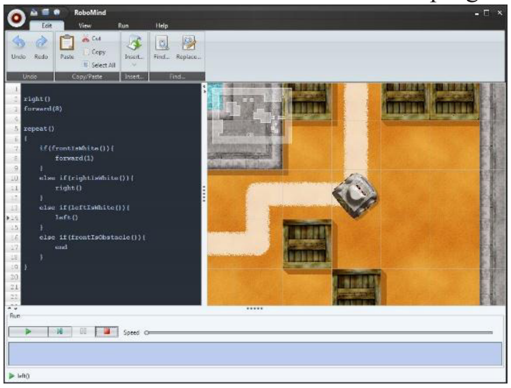
3.1. RoboMind

RoboMind is very simple educational programming language used for getting users familiar with the basics of computer science. It introduces popular programming techniques, and helps gain an insight into areas such as robotics and artificial intelligence. These skills are gained by creating programs for a robot.

The robot is capable of performing several actions. It can drive, look around, move items and paint. This can all be done in different environments that are made up of blocks.

RoboMind uses an imperative/procedural programming language. The language consist of a number of basic instructions to control the robot, repetition loops, conditional if-then statements, and it is also possible to define custom instructions by creating procedures. Procedures are allowed to be defined recursively.

RoboMind is meant to serve as a first introduction to automation and programming without prerequisites.



```
right()
forward(8)

repeat()
{
    if(frontIsWhite()){
        forward(1)
    }
    else if(rightIsWhite()){
        right()
    }
    else if(leftIsWhite()){
        left()
    }
    else if(frontIsObstacle()){
        end
    }
}
```

Fig. 1. Example of Robomind program

Difficulty level can vary, depending on the intended audience. In primary education pupils can get acquainted to writing commands to navigate the robot through its environment, on high school programming structures get more attention and universities focus on the theoretical aspects of automation theory like Turing machines.

3.2. NXT-G

NXT-G is a graphical programming environment that comes with the LEGO NXT Mindstorms robot, which is also used in one of our teaching phases. Programs in NXT-G are created with careful construction of blocks and wires, and can be used for real-world programming (Kelly, 2010). Each block corresponds to some sort of command regarding movement, sensors, logical operation etc. Most of those blocks have some additional parameters that can also be adjusted like the power of the motor or sensor sensitivity. NXT-G also supports complex functions like parallel threads and remote control. The language supports virtual instruments for a variety of sensors/components. Main advantage of this software is that is used in pair with an actual robot, so the users can almost instantly see their program in action.

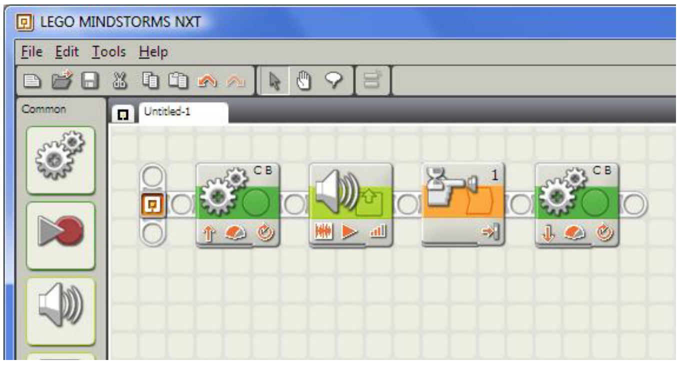


Fig. 2. Simple NXT-G program

3.3. Scratch

Scratch is a programming language learning environment created with an idea to enable beginners to quickly create programs without having to learn how to write them syntactically correct. It is intended to motivate further learning through playfully experimenting and creating projects, such as interactive animations, games, etc. (Maloney, Resnick, Rusk, Silverman & Eastmond, 2010).

Scratch features a graphical user interface which prevents the need to memorize the programming language, and instead allows for users to explore its different functions.

Main program is made from code fragments (called "blocks") that can be dragged onto the scripts area to make programs. Blocks are organized into different categories such as movement, looks, control, sensing, etc. for easier use. Different kinds of blocks also have different colors and shapes.

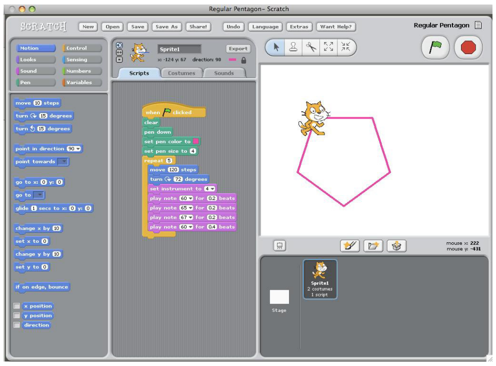

```
when clicked
clear
pen down
set pen color to
set pen size to 4
repeat 5
  move 120 steps
  turn 72 degrees
  set instrument to 4
  play note 60 for 0.2 beats
  play note 65 for 0.2 beats
  play note 67 for 0.2 beats
  play note 60 for 0.4 beats
```

Fig. 3. Scratch interface and simple program.

It can be noticed that all of them have special focus on visual presentation. That is considered quite important, especially considering the age of users it is intended for. Because of that, all aspects of teaching approach described in this paper also have strong emphasis on visual presentation, thus improving the user (children's) experience.

Another important characteristic is usability or "ease-of-use" of the application. Again, all above applications are fairly easy to use and have very simple set of basic commands. Some of them (NXT-G and Scratch) have the option of making custom, more complex commands. It should be taken into account that teaching method described in this paper is intended for primary school children and teaching a few basic concepts, therefore it would be sufficient to keep the set of commands minimal so that children would not be burdened with excessive set of data that they would have to memorize. It is more important to teach them the process of logical thinking and problem solving (Adams, Kaczmarczyk, Picton, & Demian, 2011). Once they have developed that kind of knowledge they can easily apply it to the all other aspects of education as well as real life problems.

Some of these applications are used in college for teaching students whose primary field of study is not computer science. During discussions with professors using those applications it was pointed that the majority of students are trying to solve given problems using the "trial and error" technique. Main problem is, that in most cases, that is the only technique they are using, without ever trying to logically analyze a given problem and produce an adequate solution. This is exactly the situation in which the approach described in this paper would be quite useful, so in next section that approach will be presented in detail.

4. SOLUTION

To counter problems with teaching programming described before, we are proposing an approach to that should help elementary school children in acquiring basic programming concepts. This approach consist of four different phases. Each individual phase is described in detail in following paragraphs.

As already mentioned, best results are expected if the learning process is hidden from children and instead they are presented with something that resembles some kind of a game. In our case, we decided to use the LEGO Mindstorms robot and NetLogo simulated environment to present the children with a simple problem of navigating a robot through a maze in order to find a hidden prize. This way children would be interested in a robot and simulation (presented as computer game), but they simultaneously adopt some specific knowledge and concepts.

4.1. First phase

First phase of our teaching process is done by conducting a verbal interview with the children. This phase corresponds with the lowest level of learning and its intended goal should be for children to familiarize themselves with some basic concepts - What is a robot? How is he controlled? Can you change his behavior and how? For best results this phase should be conducted with smaller groups of children (5-6) so that teacher can talk to each of them individually if necessary if they are not familiar with those above concepts. At the end of this phase they should be aware that they have the ability to control the robot and they are the ones that should provide him with the instructions required to solve a given problem.

4.2. Second phase

In second learning phase, children should be introduced with the rules of the system, what commands are made available to them and what is the goal of the game. They should comprehend that this specific problem has a set of rules and limitations, and in order to find a solution they should comply with these rules. If successful, they will transfer this knowledge to the other problems and learn that whenever presented with some problem they should first define the goal of the problem and examine all the options available for solving that specific problem.

In this phase, we decided to use “unplugged” programming (Bell, Alexander, Freeman & Grimley, 2009). We

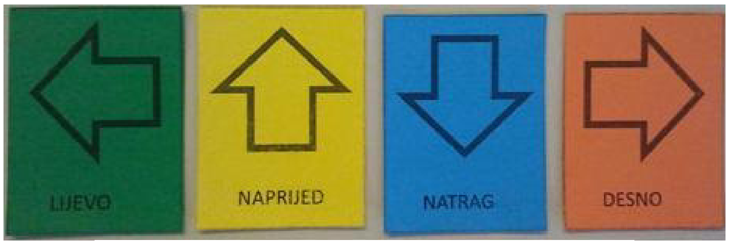


Fig. 4. Colour coded cards used in second phase.

have prepared a set of marked and color coded cards, each one representing a single command available to the robot for execution. Cards are shown in Fig.4. They are 4 basic commands available to children (up, down, left and right), each one corresponding to the movement of a robot in discrete space of a maze. Children are presented with the multiple cards of each type and their task is, using those cards, to define a sequence of commands which, in their opinion, would lead the robot to the goal of the maze. By doing this, we are preventing them of abusing the "trial and error" method and instead encouraging them to visualize the problem and use their imagination in deriving the acceptable solution (Edwards, 2004).

This approach should demonstrate them that programming (and problem solving generally) consists of series of simpler steps that are combined into one greater program. It could also help them to notice that some steps (or a series of steps) can be repeated thus learning the definition and usage of "loops". Unlike previous phase, this phase is practical, not only theoretical as children themselves must try and produce a solution to the given problem. This is necessary because in order to learn some concepts it is not just enough to present the children with those concepts. Programming is not learned simply by knowing all the syntax and semantic rules, but it requires an active knowledge of different concepts which is achieved only by applying those concepts in specific, real world problems.

As this approach is focused on primary school children, they are not burdened with specify programming language and syntax learning. Instead, they are introduced to few simple concepts corresponding to the robot movement. These concepts should be fairly easy for them to comprehend as these are real world concepts that they already know, only applied to the robot instead of humans.

4.3. Third phase

Third phase consists of displaying the robot in simulated environment using NetLogo, a multi-agent programmable modeling environment (Sklar, 2007). Robot is displayed as single agent. Simulated environment is depicted as a maze through which the robot must navigate in order to reach its goal (presented as the cake within the simulation). In addition to displaying the world, simulation includes controls for moving the agent that are corresponding to the cards used in previous phase. During this phase, children should make a logical connection between the physical (real life) robot and the agent presented in the simulated environment. This is very important as that introduces them to the basic level of abstraction.

Children should show their cards and each command should be entered into the simulation. Simulation is designed so that the each command is immediately executed but it also keeps a record of all given commands so

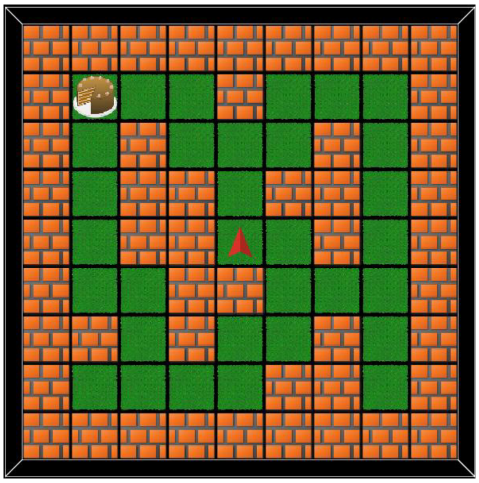


Fig. 5. Simulated world depicted within NetLogo simulated environment.

that they can later be reviewed. This enables children to see the results of each action, and also the completed set of commands i.e. their first computer program. During the input of commands into the simulation they should be allowed to make corrections to commands that haven't yet been executed if they think that is necessary. This should demonstrate them that there are often multiple ways of solving one problem and encourage them to make simple modifications to their programs as well as trying to find the best possible solution.

It should also familiarize them with concept of debugging i.e. testing the solution in controlled environment before marking it as final.

4.4. Fourth phase

Final phase consists of executing the program developed in previous phase. NetLogo simulation keeps records of all entered commands and at the begging of this phase those command are send to the physical robot for execution. As mentioned, a LEGO NXT Mindstorms robot is used and set in a maze that corresponds to the maze depicted in the NetLogo simulation. Communication between NetLogo and Mindstorms robot is done via Bluetooth, but that is not important for the teaching process alone. Goal of this phase is to further establish the link between real and simulated world, this time demonstrating to children that the actions they previously made have an impact in the real world and that their programs are put in use. This should hopefully cause immersion during programming i.e. children would be more likely to inadvertently put themselves into the role of a robot they are programming. By doing that, they would have greater insight about the capabilities and limitations of their program and that should result with better written programs.

5. EXPERIMENTAL STUDY

Preliminary experiment was conducted in elementary school with children from first grade. Children were divided in 6 separate groups, each group consisting of 5 children. There was no further division based on certain attributes like gender or previous knowledge. This was done because certain children were in need of individual attention and putting them in larger groups would prevent that. This also made communication with children easier.

During the experiment, children went through all 4 phases described in this paper, from verbal introduction to physical world demonstration of their program. Completion of those phases took approximately 40 minutes for each group which fits the time of one period in elementary school.

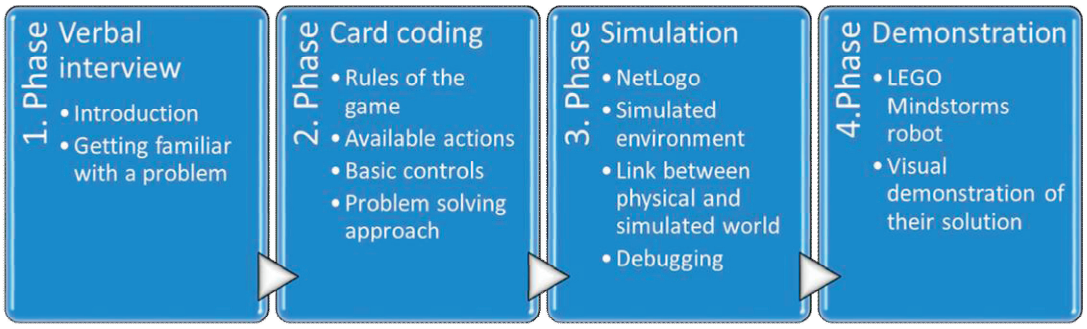


Fig. 6. Short overview of four different phases of teaching process.

Most of the children had very similar knowledge of computers. They were already familiar with basic computer concepts and have acknowledged using computers for both recreational and educational purposes. One thing we noticed is that gender does not affect their computer skills, both boys and girls where equally proficient with basic computer use. Their descriptions of the concept "robot" were very similar and fairly accurate, regarding their age.

They all described robot as metallic manlike thing that can move autonomously. All the children had clear understanding that some kind of electric power (batteries) is needed for robots normal operation. Most of them have recognized LEGO bricks as building elements of presented autonomous vehicle. They have also asked if our vehicle can transform into another shape, but they have also pointed out that we have to perform the transformation. This behavior is showing that there is a clear distinction made between cartoons and physical environment but it also shows that the expectations are big influenced by virtual worlds constructed in cartoon designer studio.

Most of them guessed that robot is controlled remotely by some sort of hand-held remote controller but they quickly grasped the concept of giving the robot a set of instructions that will be executed as soon as robot is turned on.

6. CONLUSION AND FUTURE WORK

Children reacted positively to the whole teaching process and they expressed their pleasure with the whole procedure. After reviewing all the programs that children created for navigating the robot through the maze, it was noted that only one solution was incorrect. It was also very interesting to see that most of the correct solutions were also optimal solution for that particular maze, meaning that proposed solution was quickest way from start to goal (minimal number of actions). This is interesting because children were not presented with a concept of “shortest path” or “optimal solution” but they nevertheless displayed that kind of knowledge. Taking in account their age, it is more likely that they innately possess this type of knowledge, then they somehow previously learned it from their parents or surrounding. This is something that should be tested more thoroughly in future research.

As this was only preliminary research, obtained results were encouraging but for more concrete results it will be necessary to expand the research by using much larger number of children and programming concepts. That should produce more data necessary for testing the effectiveness of this approach.

References

Adams, J., Kaczmarczyk, S., Picton, P., & Demian, P. (2011). Problem solving and creativity in engineering: conclusions of a three year project involving reusable learning objects and robots. Engineering Education: Journal of the Higher Education Academy Engineering Subject Centre, 5(2), 4-17.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. The New Zealand Journal of Applied Computing and Information Technology, 13(1), 20-29.

Edwards, S. H. (2004, March). Using software testing to move students from trial-and-error to reflection-in-action. In ACM SIGCSE Bulletin (Vol. 36, No. 1, pp. 26-30). ACM.

Kelly, J. F. (2010). Lego Mindstorms NXT-G Programming Guide. Apress.

Kirriemuir, J., & McFarlane, A. (2004). Literature review in games and learning.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. ACM Transactions on Computing Education (TOCE), 10(4), 16.

Meluso, A., Zheng, M., Spires, H. A., & Lester, J. (2012). Enhancing 5th graders’ science content knowledge and self-efficacy through game-based learning. Computers & Education, 59(2), 497-504.

Palfrey, J., & Gasser, U. (2008). Born digital: Understanding the first generation of digital natives. Basic Books.

Prensky, M. (2008). Digital game-based learning.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137-172.
Seybert, H. (2012). Internet use in households and by individuals in 2012.Eurostat Statistics in Focus, 50/2012.
Sklar, E. (2007). NetLogo, a multi-agent simulation environment. Artificial life, 13(3), 303-311.
Wadsworth, B. J., & Gray, W. M. (2004). Piaget's theory of cognitive and affective development. Pearson/A and B.
Yadin, A. (2011). Reducing the dropout rate in an introductory programming course. ACM Inroads, 2(4), 71-76.