PROGRAMMING MICROWORLDS FOR VISUALLY IMPAIRED PUPILS

Ľudmila Jašková, Mária Kaliaková [1)]

Abstract

The paper describes our research aimed at verification of the suitability of two programming environments for pupils of elementary schools with visual impairment. Our aim is to develop environments suitable for computer novices and usable for all the children – blind, partially sighted and without vision disorders. In addition, we designed a set of educational activities aimed at development of basic programming competences. For verification of the environments as well as the learning activities the design based research was used.

Keywords blind, partially sighted, environments for children, programming, learning

1. Introduction

Since September 2011, we teach informatics at a primary school for visually impaired children. There are three types of pupils – blind, partially sighted, pupils with normal vision but with some kinds of learning problems. To avoid discrimination, our effort is to use the same software with all the pupils. Of course, pupils with different impairments use appropriate assistive technology, as well. The methods of usage of computers by particular types of pupils are presented in Section 2. We have not found yet the suitable programming environment for all three types of children, as we explain in Section 3. We developed two new environments built on our experience with teaching these pupils [1]. We describe these environments in Section 4. Furthermore, learning activities for using them with children were prepared (Section 5). We put into effect a design based research aimed at evaluation of these environments and learning activities. This research and the obtained results are described in Section 6.

2. Observed pupils

In this section, we describe the observed pupils and their informatics skills. During our informatics lessons pupils are usually split into two groups:

– totally blind – children that have no usable vision and have to learn using Braille, screen readers and another non-visual means,
– partially sighted children that have a heavy visual disorder and have to use a special aid (magnifying software) to learn and pupils without vision disorders but with some kinds of learning problems.

[1] Dept. of Informatics Education, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia

In our paper, we will describe the work with two groups – the group of blind pupils and the group of partially sighted pupils. The first group consisted of four blind girls from the fifth and sixth grades (age of 10 to 12). Three of them were blind since their birth and the other one was blind since her age of six. This girl was more ambitious; she had better communication skills and better orientation skills than her blind classmates. All these girls started with computers and informatics only in the fifth grade, when they sat at computers for the first time. We performed our activities described in this paper after they all had become familiar with the keyboard and they were able to write a simple text and to use key commands for reading a text.

Figure 1 Blind child using a computer

Group of partially sighted pupils consisted of five eighth graders (age of 14) – three boys and two girls. One pupil had a low peripheral vision and one had a disorder of the color perception. Each of them needed to use some sort of magnifying software so that they could enlarge the screen 2- or 4-times. Two pupils used also a high contrast function and, consequently, the colors in the programs were changed. These pupils already used the Scratch environment for programming. They were able to edit a text without problems and they could use tables.

3. Programming environments for children

In our country, informatics as a separate specialized subject includes also the topic algorithms and problem solving [2]. This topic is always adapted with regard to the age and capability of pupils and according to them also appropriate programming environments for children are selected. There is no reason to miss this topic when teaching visually impaired pupils. There are several blind programmers in IT companies. Besides, each blind person should understand when somebody explains a way for her/him in words and should be able to walk it. It is therefore the execution of an algorithm. Curriculum for pupils with visual impairment [3] imposed to gain the basics of algorithmic thinking. Graduates from primary schools (age of 15 or 16) should understand concepts like *command*, *sequence of commands*, *loop command*, *procedure*, *parameter*, *variable*, *value*, *assignment*. They should also be able to use a formal notation for writing *sequence of commands* and interpreted them. For teaching these pupils, it is necessary to choose appropriate teaching methods and software tools.

Now, we will give an overview to programming environments and teaching methods that inspired us in development of our software and learning activities. Teaching algorithms and problem solving in standard class with pupils without vision disorders is usually done via software that leans heavily on visual representations (Logo, Scratch ...) [4], [5]. All these environments allow the manipulation of virtual objects with a wide variety of commands. But these educational software products are not usable, if we teach totally blind children – they work with computers using a screen reader and the only information they can work with is the text and sound. We considered the usage of *audio programming language* (APL) for the blind developed in Chille [6], but this language was not available for us. Moreover, the pilot testing with three blind novice programming learners, ages 17 to 20, showed that programming in this environment was too abstract even for them. We agree with Futschek [7], [8] that programming is an abstract activity that can be managed successfully only if the child has enough experience with the control of specific tangible objects. The author had in mind that children themselves perform a simple program by *manipulating tangible objects*. But there are also so-called *tangible programming languages*. Horn [9] describes two tangible programming languages, *Tern* and *Quetzal*, which he develops with his colleagues at the Tufts University in Medford (Oregon, USA). The Tern language allows children to create a program for control of the virtual robot Karel on a computer screen and the language Quetzal for control of the LEGO Mindstorms^{TM} robot. In both cases, the program is created from tangible elements beyond the computer. The finished program is scanned with the camera and transformed into an iconic language that is on the computer. Then it is executed by the robot. The authors create these tangible languages for sighted children to facilitate their collaborative development of programs in the group. We find the language Quetzal suitable for our students (blind and partially sighted), as they could create a program by hands and they could also check the program execution by the robot by touch. Unfortunately, both languages are not commercially available yet, because their use has been restricted almost entirely to laboratory and research settings.

We wanted to enable our pupils to learn programming in a constructivist manner in terms of Papert [10]. He believes that *knowledge is built by the learner* and that doing so through an actual construction makes it more genuine [11].

We found another source of inspiration in *computer science activities without using a computer* (CS Unplugged) developed by the Canterbury Computer Science education research group. Bell said that “These activities generally have a strong *kinesthetic component* and take a constructivist approach: students are given enough clues to enable them to work out principles for themselves” [12].

4. Programming environments for visually impaired children

We have not found any suitable programming environment that would be text-based and friendly to blind beginners who have no previous experience with programming. In this situation, we have decided to teach the basic algorithms using a Bee-Bot toy (Figure 2) [13].

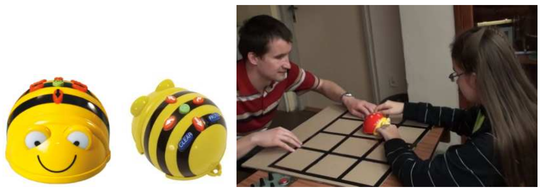

Figure 2 Robotic toy Bee-Bot and special Bee-Bot mat for blind children

4.1. Bee-Bot toy

We used the Bee-Bot as a real tangible object. It has a limited number of commands that are easy to understand, so it is suitable to learn the basic concepts of algorithmic thinking – *commands*, *sequence of commands* (arrow keys on Bee-Bot). We confirmed this at our informatics lessons [1]. But we needed to go further towards the virtual learning environment and the world of computer programming.

4.2. Tactile table and a simple text editor

Our virtual learning environment consisted of a tactile table, sticky rubber and simple text editor. We used a plastic tactile table instead of the cardboard mat. The table had 10x10 fields; each field has one square centimeter. We marked the way for our virtual bee with a sticky rubber (Fig. 3). We used a wooden figurine instead of the bee. We marked some fields on the table with a rubber and the task for pupils was to write instructions in text for the bee, to visit all the marked fields. The set of allowed commands was: *left*, *right*, *forward*, *backward*. Pupils could write in short: *LT*, *RT*, *FW*, and *BW*. They could also write a number in front of the command, indicating the number of its repetitions. For example, they could use the command *3FW* instead of *forward*, *forward*, *forward*. Such a number could also be written in front of parentheses embracing a sequence of commands. Basically, it was programming without a PC. Pupils used the text editor for writing a sequence of commands, but computer did not execute these commands. The only feedback was from the teacher. This environment was more abstract than programming of the Bee-Bot toy. But it had the following advantages.

- Possibility to fix bugs in the program. It was possible to change only the defective part of the program, in contrast with the Bee-Bot programming, where it was necessary to retype all the sequence of commands.
- Possibility to use the loop construct with a fixed number of repetitions.
- Possibility to solve the different type of tasks – playing a given algorithm, finding bugs in the program, finding the better solution. These skills belong to higher levels of Bloom taxonomy (analyze) [14].

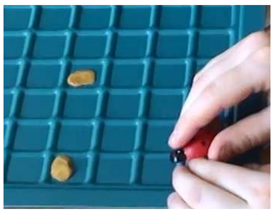
Figure 3 Simulating the program execution on a tactile table

After mastering the work in this environment, the pupils were ready for programming in the virtual programming environment, allowing the control of the virtual object moving on the computer screen.

4.3. Programming environment Lady Beetle

Student Ľuboslava Bečvarová [15] developed, in cooperation with us, the application Lady Beetle. The application has the following features.

- This tool allows the control of the virtual object moving on the virtual table in the computer. While the object is moving according to the instructions, coordinates of the passed fields are pronounced verbally to the blind user so that he/she has the feedback.
- It contains a small set of commands in a vernacular language. The basic commands are *forward*, *left*, and *right*. Furthermore, it is possible to use the *loop command* with a fixed number of repetitions and the *conditional command*. This allows the program branching, depending on whether the lady beetle is on the edge of the playing area, or if there is any object (flower, house) on the current field.

– The programming interface contains a command list introduced to the user to select the desired command. There is no need to type them.

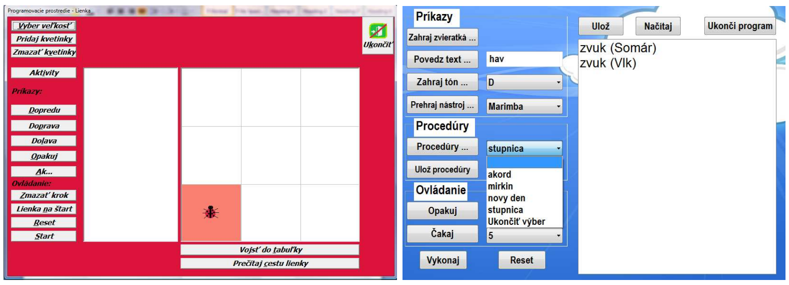

Figure 4 Lady Beetle environment (on the left) and World of Sounds environment (on the right)

Control of the virtual object that exists only on a computer screen is quite abstract and requires good orientation skills in the plane. Therefore, we decided to develop another type of programming environment in cooperation with another student, Jana Sucháneková [16]. Working in this environment does not require imagining the movement of the object in the plane.

4.4. Programming environment World of sounds

Programming environment Word of sounds offers a very simple audio programming language and has the following features.
– This tool allows to program a sequence of sounds – either it plays the selected audio files or pronounces the entered text.
– The programming interface contains a command list introduced to the user to select the desired commands. There is no need to type them.
– The basic commands are used to play different sounds or to pronounce the entered text. In addition, the programming language includes commands that lead students to understand more advanced concepts like *cycle* and *procedure*.

We hope that through the variety of sounds, programming will be funnier for the blind children than if they used only a synthetic speech. Požár [17] claimed that blind children with low stimulation have a reduced curiosity instinct. It is clear that when listening to a monotonous synthetic speech, blind pupils get tired quickly and their attention reduces. Application controls (buttons, lists of commands) use the recorded sound files, i.e. natural speech. The program contains the menu of commands to play different sounds that pupils can use when creating their programs. These are sounds of animals, different tones and musical instruments. There is a statement *SayText* allowing to express any text entered in the textbox. It is pronounced through a screen reader.

5. Learning activities

Now we describe the educational activities that we used for teaching programming in environments mentioned above. We can say that our pupils gradually passed through the following three phases.
1. Programming the Bee-Bot toy
2. Writing programs and their simulation using a tactile table and simple text editor
3. Real computer programming
   – Programming a virtual object moving along the computer screen in the Lady Beetle program
   – Programming a sequence of sounds in the World of Sounds program.

5.1. Activities with the Bee-Bot toy

At the beginning, we explained how to control the bee and what is its environment. Pupils explored both by their hands. As a motivation we used a story about the bee that is flying from her house to the nearest flower, then to the next one and after she visits all the flowers in the meadow, she will return back home. Initially we let the pupils solve very easy tasks, so that they understand the meaning of basic commands (move the bee few fields forward and turn right or left, move her back home). When pupils had got familiar with the basic commands, we gave them more difficult tasks (move the bee alongside the border, visit all fields on the mat, move the bee alongside the letter O, L, U). Whereas we could use more Bee-Bot toys and each pupil could use its own one, we did a race [1].

5.2. Activities with the tactile table

First we have explained the allowed text commands to control the bee. Initial tasks were the same as the initial tasks with the Bee-Bot toy. When pupils got to be familiar with the commands, they were able to solve more complicated tasks (move the bee alongside the square or rectangle, move her diagonally, visit three randomly selected fields) [1].

5.3. Activities in the program Lady Beetle

As we have mentioned above, the children first performed activities with the Bee-Bot, where they constructed algorithms using the arrow keys. Then activities on a tactile table followed, where commands were represented by text. We explained the pupils that, in the program Lady Beetle, the same commands will be used for moving the lady beetle on the computer screen. We used the tactile table as well to simulate the situation on the screen. Initially, we let the pupils solve very easy tasks, so that they understand the meaning of the basic commands. The first task was to get the lady beetle one field forward. The next task required to use also the command right (Figure 5 on the left) and then also the command left (Figure 5 on the right).

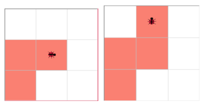
Figure 5 Program output in tasks aimed at training of the basic commands

The task when the lady beetle is flying above the bed of flowers in the shape of a square is appropriate as an introduction to the loop cycle. Pupils already solved the same task with the Bee-Bot toy and with a tactile table. Pupils can solve this task without the loop command first. Then they have to find out which part of the algorithm is repeated and how many times. We can tell them that they can put repeated sequence of commands into the loop structure and to place the number of repetitions at the beginning of the repeat command (Figure 6).

```
Repeat 4 times
  Forward
  Left
End
```

Fig. 6 Program with cycle

Other tasks for the use of the loop command are in Figure 7. The goal is to get the lady beetle home.

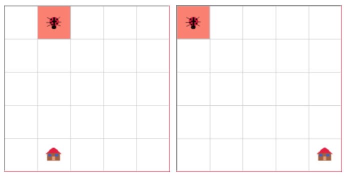
Figure 7 The task is to create the shortest program for getting the lady beetle home

5.4. Activities in the program World of sounds

Our pupils already edited audio files in Audacity [18] application. They recorded sounds and created audio stories by combining multiple recorded sounds. We introduced the application World of sounds as a similar application to Audacity enabling creation of audio stories but in a different manner. We can use different commands for playing sounds, like *PlaySound*, *SayText*, *PlayTone*, *PlayInstrument*. Pupils can try how each command works. Then they solve tasks in which it is needed to combine different commands. Pupils create story about visiting the farm.

```
SayText (I met these animals on the farm)
SayText (donkey)
PlaySound (donkey)
SayText(cat)
PlaySound(cat)
SayText(and a dog.)
PlaySound(dog)
```

Figure 8 Possible solution of the task mentioned above

We can use the command *PlayTone* for playing a simple melody. As an introduction to the loop command, we can tell the pupils a story about a disobedient pupil Joey. He was cheating on a test at school and as a punishment he had to say ten times: "I'll never cheat on the test." Joey was lazy and would rather write a program that said it instead of him. As it appeared at that program? Pupils can solve it without using a cycle first and then using a cycle (Figure 9). Alternatively, pupils can create a program which will play some sequence of tones repeated several times.

```
Repeat 10 times
  SayText(I'll never cheat on the test.)
end
```

Figure 9 Program with cycle

6. Evaluation of programs and our learning activities

For evaluation of programs and our learning activities, a design based research [19], [20] was used. Thus, in several iterations, we developed the programming environments and educational activities mentioned above and we observed how we managed to achieve educational objectives using them. We searched for answers to the following research questions.

– What features should a programming environment for blind and partially sighted children have?
– What learning activities are suitable for the development of programming competencies of the observed pupils?
– Which informatics competencies can we build and develop using our environments?

Created programs and educational activities were used with a group of blind and visually impaired students described in Section 2. There were several independent observers on lessons. They took notes and we recorded the lessons by the video camera, as well. Subsequently, we analyzed the recorded and written material. Now we describe our observations for each group – a group of blind and a group of partially sighted pupils.

With the group of blind pupils the above activities at nine lessons were performed. We performed the Bee-Bot activities (see Section 5.1) at two lessons and activities on a tactile table (see Section 5.2) were performed at three lessons. We encountered the following problems during these lessons.

- Confusion of commands: *left* and *right*, especially if the bee was turned in a different direction as the learner's head was.
- Problems requiring many turns and long paths are difficult, because pupils became lost in programs and on tables as well.
- Problem occurs when learners have to decide how to find the way between two marks which are neither in the same row nor in the same column.
- We discovered that our learners prefer writing the same sequence of commands instead of using a *loop command*. This means that this concept needs more practice.

Then we performed two lessons with the Lady Beetle program and two lessons with the World of Sounds program. Pupils liked activities in both programs. They worked with programs without problems. In a short time they were able to manage most of the necessary actions by using keyboard shortcuts. While using the program Lady Beetle, we used the tactile table to simulate the situation on the screen. The pupil, who lost her sight later in life, had a better spatial orientation and better abstract thinking then her schoolmates blind since their birth. Some problems occurred while they were simulating the program on the table.

- Pupils did not realize that when the lady beetle moves one field forward, only the line or column number changes and not both coordinates.
- Pupils did not realize that when the lady beetle turns right or left, she does not make a step in that direction automatically.

In program the World of sounds, blind pupils appreciated that the application uses synthetic speech and "live" sounds as well. They enjoyed examining what sounds application provides and they enjoyed creating stories combining text and different sounds. First they gladly created stories in our award, but over time they liked to invent their own assignment for the story creation. While working with this application, we did not register any differences in success among girls, who have been blind since birth, and a pupil who lost her sight later in life. We have discovered that our learners prefer to write the same sequence of commands instead of using a loop command.

With group of partially sighted pupils the above activities were performed at four lessons. We performed two lessons with the Bee-Bot toy. Majority of pupils were able to solve the tasks with the Bee-Bot immediately, even without simulating its movement across the mat. Only the pupil with color perception disorder had problems with orientation on the mat and she probably had poor spatial orientation [1], [5]. Then we used the program *World of Sounds* at one lesson. Pupils solved the tasks mentioned in Section 5.4. They preferred to control the program by a mouse (clicking on buttons) to using keyboard shortcuts in both programs. They had no problems to control the program and to solve the tasks, as well. Partially sighted children enjoyed creating audio stories. They enjoyed that the computer obediently executed their commands. Some students were very creative and created long and funny audio stories. They were able to solve the tasks requiring the loop command without difficulty, but when they could choose whether to use loop command or to write the same sequence of commands several times, they preferred to avoid the loop command. Then we used the Lady Beetle program at one lesson. Pupils quickly understood what to do.

Severely visually impaired pupils appreciated when the screen reader told them the coordinates of the passed fields by the lady beetle. This was helpful for them, because they could not see all the playing area at once. This feature was useful also to train the navigation in the tables and coordinate system. Pupils solved the tasks mentioned in Section 5.3. When pupils got to be familiar with the basic commands, they preferred to set the biggest playing area (10x10 fields) with five flowers in random fields and solved their own tasks. Lady beetle should visit all the fields containing flowers and return back to the starting field. Pupils really enjoyed solving such tasks. Together with making programs they have learned to find a bug in the program and to fix it.

Now we *answer the questions* written at the beginning of Section 6. We found that both environments work well with screen readers, and a magnifying program. In the Lady Beetle program occasionally the screen reader does not read the correct sequence of coordinates and also the magnification program does not show the needed area of the screen. These bugs in the program should be fixed. As regards the usability, children can easily and intuitively control both programs by a keyboard and by mouse, as well. Captions on the buttons are understandable and the buttons are in a relevant tab order. It is easy to remember the keyboard shortcuts. Learning activities were appropriate for our pupils. In their implementation, however, the individual abilities of students must be taken into account. Particularly in activities requiring a certain degree of spatial and visual imagination, it is necessary to use different methods with pupils blind from birth, who have not developed visual memory and imagination, and other methods with pupils who have lost their sight after development of visual imagination. We recommend to simulate the situation on the screen using the tactile table for both types of students. But the first type of students requires more time and easier tasks (smaller playing area with fewer flowers). Partially sighted pupils can solve more complicated tasks than the blind ones and do not need to be told the coordinates of the passed fields, because they have a visual feedback. The basic commands are clear, but, initially, students expect a different interpretation of the commands right and left in the program Lady Beetle. Probably, it would be preferable for the blind pupils, if these commands do not turn the lady beetle, but move her one field in the appropriate direction. As regards the building of algorithmic competencies, we assumed that students understand the meaning of concepts *command*, *the sequence of commands*, *program*. Pupils prefer retyping the same sequence of commands to using a cycle. Output of the program created in the Lady Beetle environment (the sequence of coordinates of the passed fields) was understandable for the blind pupils, but they would appreciate special sound effects assigned to some events (e.g. occurrence of flower or house on the current field). The Lady Beetle application seems to be suitable not only for the development of algorithmic thinking, but it is a good tool to practice the work with tables and orientation on the desktop. The World of Sounds application in turn, in addition to the development of algorithmic skills, is suitable to train working with text and working with controls of the dialog box. Pupils learn these skills in a funny way. Moreover, creating various audio stories contributes to the development of child creativity.

7. Conclusions

Our environments are physical micro-worlds [11] which help visually impaired learners train algorithmic thinking and help them understand the very basic algorithmic concepts. Besides the fact that children train their orientation in the space, they solve logic problems and make their first steps in environments similar to a programming environment. Our learning activities were suitable for our learners and we hope that they will be useful also for other teachers and their pupils (sighted and blind as well). The Lady Beetle environment and World of Sounds environment are not perfect. We did just the first step in development of appropriate universal programming environments available for children. We face a number of challenges. We still have not found a suitable way to introduce more complicated algorithmic concepts such as a conditional command, procedure, or variable. Our task in the near future will be to add these features to our environments and learning activities.

References

[1] Ľ. Jašková, "Blind Pupils Begin to Solve Algorithmic Problems," in *Sixth International Conference on Informatics in Schools*, I. Diethelm and R. T. Mittermeir, Ed., LNCS 7780, Heidelberg, Germany: Springer , 2013, pp. 68-79.

[2] *National Educational Programme. Informatics* (in Slovak). [Online]. Available: http://www.statpedu.sk/files/documents/svp/2stzs/isced2/vzdelavacie_ oblasti/informatika_isced2.pdf. [Accessed 13 3 2014].

[3] *Educational Programme for Cildren and Pupils with the Visual Impairment* (in Slovak). [Online]. Available: http://www.statpedu.sk/files/documents/svp /vp_pre_zz/vp_zrakovo/vp_zp_%20isced_1_2_3_5.pdf. [Accessed 13 3 2014].

[4] A. Blaho *et al.*, *Imagine Logo Primary workbook*, Cambridge, Great Britain: Logotron, 2004.

[5] M. Kabátová *et al.*, "Fostering creativity through programming – Scratch workshop," in *Fifth International Conference on Informatics in Schools*, I. Kalaš and R.T. Mittermeir, Ed., LNCS, vol. 7013, Berlin Heidelberg , Germany: Springer, 2011, pp. 155-164.

[6] J. Sánchez *et al.*, "Blind learners programming through audio, " in *CHI EA '05,extended abstracts on Human factors in computing systems*, New York: ACM, 2005, pp. 1769-1772.

[7] G. Futschek *et al.*, "Developing Algorithmic Thinking by Inventing and Playing Algorithms," in *Constructionism*, J. Clayson and I. Kalas, Eds., Paris, France, 2010. pp. 16-20.

[8] G. Futschek *et al.*, "Learning Algorithmic Thinking with Tangible Objects Eases Transition to Computer Programming," in *Fifth International Conference on Informatics in Schools*, I. Kalaš and R.T. Mittermeir, Ed., LNCS, vol. 7013, Berlin Heidelberg , Germany: Springer, 2011, pp. 155-164.

[9] M. Horn *et al.*, "Designing Tangible Programming Languages for Classroom Use, " in *First International Conference on Tangible and Embedded Interaction*, New York, NY: ACM, 2007, pp. 159-162.. [Online]. Available: http://www.cs.tufts.edu/~jacob/papers/tei07.horn.pdf. [Accessed 12 10 2013].

[10] S. Papert, *Mindstorms: Children, computers, and powerful ideas.* 2nd ed., New York: Basic Books, 1993.

[11] S. Papert, "What is logo? And who needs it?," in *Logo Philosophy and Implementation*. Logo Computer Systems Inc., 1999.

[12] T. Bell *et al.*, "Computer science unplugged: School students doing real computing without computers," *The NZ Journal of applied computing and information technology*, no. 13, 2009, pp. 20-29.

[13] M. Kabátová *et al.*, "Robotic Activities for Visually Impaired Secondary School Children" in *Third International Workshop, "Teaching Robotics, Teaching with Robotics", Integrating Robotics in School Curriculum*, Riva del Garda, Italy, 2012.

[14] B. S. Bloom, *Taxonomy of Educational Objectives.* Boston, MA: Allyn and Bacon. 1956.

[15] Ľ. Bečvarová, "Programming Environment for Children with Visual Impairment – Programming the Robot," Bc. Theses (in Slovak), Dept. of Inf. Edu., Comenius Univ., Bratislava, Slovakia, 2013.

[16] J. Sucháneková, "Programming Environment for Children with Visual Impairment – World of Sounds", Bc Theses (in Slovak), , Dept. of Inf. Edu., Comenius Univ., Bratislava, Slovakia, 2013.

[17] L. Požár, *Patopsychology – Psychology of Individuals with Various Kinds of Impairment.* (in Slovak). Bratislava, Slovakia: MABAG, 2003.

[18] Audacity. [Online]. Available: http://audacity.sourceforge.net. [Accessed 13 3 2014].

[19] I. Kalaš *et al.*, *Metamorphoses of Schools in the Digital Age*. (in Slovak), Bratislava, Slovakia: SPN - Mladé letá, 2013.

[20] The Design Based Research Collective. [Online]. Available: http://www.designbasedresearch.org. [Accessed 13 3 2014].