ON-LINE LEARNING ENVIRONMENT FOR TEACHING THE LOGO PROGRAMMING LANGUAGE

Sanja Angelovska
Faculty of Computer Science and Engineering
Skopje, Macedonia
Email: sanja_angelovska@hotmail.com

Dejan Gjorgjevikj
Faculty of Computer Science and Engineering
Skopje, Macedonia
Email: dejan.gjorgjevikj@finki.ukim.mk

ABSTRACT

Learning the first steps in programming is one of the key activities in the area of educational computing. Although a lot of different programming languages and environments have arisen in the past years, many of the modern ones being completely visual, the LOGO programming language is still considered one of the most elaborate and most widely used for learning the first steps in programming for the children in elementary education. In this paper we present an educational programming environment based on the LOGO programming language implemented entirely as a web application. Using only a web browser the users can access the web application where they can learn the LOGO programming language by writing LOGO programs, executing them and instantly observing the output of the execution in the browser. The code editor and the interpretation of the LOGO code are implemented in JavaScript and the output is drawn on HTML5 canvas element so that complete processing is performed client side. The code editor is aware of the LOGO syntax and provides syntax highlighting, auto completion and programming help in the form of tooltips. The web page promotes the "learning by doing" approach to LOGO learning. It contains explanations of the basic LOGO instructions and set of predefined tasks for user exercise. The tasks are in the form of target drawing and the user is requested to write the LOGO program that will produce the requested drawing. The system also keeps a record of the user activities and his progress.

I. INTRODUCTION

Today, with the modern fast pace of living, computers and technology take very important part of everyday life. The information technologies have left a great impact on many aspects of learning, education and problem solving. They can be very useful in education for people of every age, even for children in lower grades, especially because nowadays children from very early childhood are familiar with the use of computers, mostly from playing computer games adapted for their age. From that point of view, we can say that computers can be used effectively for education purposes, through applications and methods that are adapted for the convenient age of the users. At the same time, the used methods have to be attractive enough for the users, so that they would be interested in using them. One such method is the programming language LOGO.

The LOGO programming language appeared in the late 1960s, when computers were not that popular yet, so in that time the idea for using this method in education was quite radical. It was created by Daniel G. Bobrow, Wally Feurzeig, Seymour Papert and Cynthia Solomon as an adaptation of the programming language LISP [1]. Today there are over 30 adaptations and variations of the programming language LOGO. The main idea behind the creation of this language was to help children in education particularly in learning mathematics, geometry, understanding variables and learning fundamental concepts of programming [2, 3, 4]. The LOGO programming environment is considered to have great potential for introducing children to many of the central concepts involved in programming and problem solving [2].

Aside programming, using LOGO assignments can help children in understanding geometry and the logic of drawing lines and complex shapes, so they would have a better perception of the space around them and better sense for orientation [5]. Besides these fundamental benefits, that are proven to be effective in practice, there are many other contributions from the LOGO language. Although easily accepted by children having first contact with simple programming assignments, beyond its early steps LOGO can be cognitively pretty complex.

LOGO language is very flexible in a sense that allows students to accomplish the same goals i.e. draw the same shapes or lines using many different combinations of commands, thus their creativity can be improved while they are learning the commands and trying to solve the given tasks. Also the users improve their thinking and logical reasoning skills that could be useful in many other areas for solving different kinds of problems [1].

If we analyse human activity we can conclude that the problem solving activity of human beings comes from the logic capabilities and creativity. Humans constantly have some goals and aims that they tend to achieve, but have not accomplished them yet. Those who are systematic in achieving goals know the importance of the systematic approach to problems and the importance of being able to

document and divide the steps in the approaches for solving problems. They have procedural outlook and attempt to identify sets of steps, which constitutes the solution of problems.

II. THE LOGO PROGRAMMING LANGUAGE

LOGO is dynamically typed multi-paradigm programming language designed primarily as a learning instrument. Its main feature is the "turtle", which is a kind of robot that understands LOGO procedures and executes them by drawing suitable lines and shapes on the screen. Turtle object is often represented as a triangle or with an image of a turtle. The use of this turtle allows immediate visual feedback of the commands. The turtle has a pencil attached to its body and draws the appropriate graphics according to the given LOGO commands.

Programming language LOGO has a very simple syntax that can be easily understood and accepted by children [6, 7, 8]. One of the key features of the LOGO language is the interactivity. There are some versions of the language that compile, but mostly it is implemented as an interpreted language, so that whenever the user tries to execute the commands, he/she gets immediate visual feedback for the given command if the syntax is correct or simple error messages if there was an error in the commands. This method could be very useful for children who practice the commands, to understand the motion of the turtle by imagining what they would do if they were the turtle and had to move according to the commands. Using the LOGO commands for path activities would help children to get a better perception about space and to be more aware of the two-dimensional shapes.

III. ON-LINE LOGO LEARNING ENVIRONMENT IMPLEMENTATION

The LOGO learning environment was designed to demonstrate the benefits of this language as a useful method for educational purpose on one hand, and the systematic and procedural approach in solving problems, on the other hand. It is implemented as a web application meant to be accessed on-line using a web browser. The application consists of two pages: user page and admin page. The admin page is accessible by the administrator who is responsible for managing the tasks that are given to the users for exercise (Fig. 1). The application is developed in Visual Studio 2010 as an ASP.NET Web Forms application using HTML5 elements and JavaScript. MS SQL Server 2008 Express is used on the server side as a database for storing the predefined tasks and the user's progress data.

The user interface is designed to be relatively simple and easy to use, which is important, especially considering the fact that the expected users would be children of the lower grades. Thus, the design is colorful and adapted to the age of the elementary school children.

Figure 1: Preview of admin page

The user page consists: two panels used for drawing the turtle graphics that are HTML5 canvas elements; a text editor for entering commands adapted for the LOGO language; a history panel where user can see all attempts and executed commands; and a table that shows all LOGO commands that can be used with a brief description of their meaning and usage (Fig. 2). The commands entered in the editor can be executed and their output in a form of drawing drawn by the turtle observed immediately on the left panel.

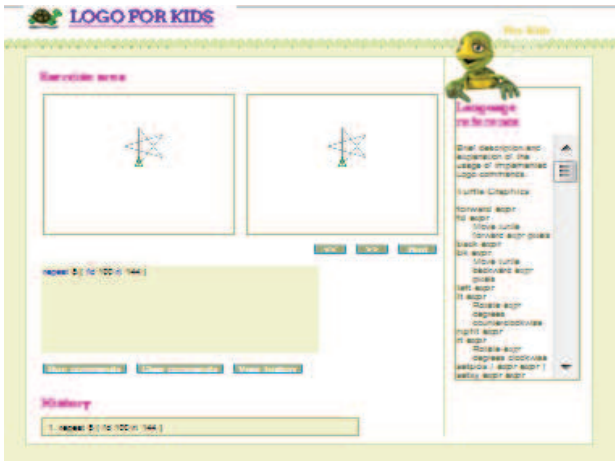

Figure 2: Preview of user page

The tasks are given to the student as a drawing in the right panel and he is challenged to write the correct commands in the editor, in order to get the same visual output in the left panel. Tasks are ordered by their complexity, which means that the first task given to the user is the simplest one and can be solved with few commands. The following tasks are with gradually increasing degree of complexity and the succeeding requested shapes contain the shapes of the previous tasks, so the student has to solve the previous assignments i.e. to learn

commands used in previous tasks to be able to solve the more complex ones.

Students can select the tasks with the buttons for navigation - Previous and Next (Fig. 3) and they try to write the adequate commands to draw the same shapes. If unable to solve the problem by himself, the student can request to see the solution of the task by clicking the Hint button. This can be very useful in the learning process because the student can see how the shape is drawn and then try to redraw it using the same commands. If a new command is used in the solution, he can see the explanation and the correct usage of that command in the language reference table and try to implement it in his own solution for drawing the shape. Also, even if the user has drawn the same shape as the one given in the task, he can see the original commands used for drawing the shape and can compare them with the commands he used in his solution. That means that the student can learn different approaches and combination of commands that can be used for drawing the same shape.

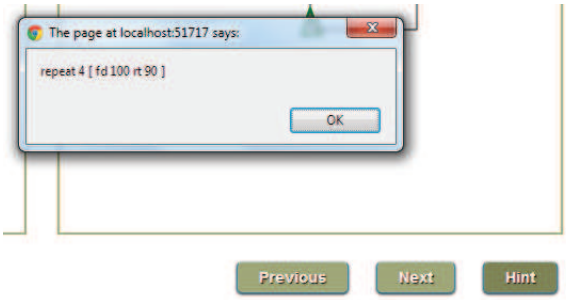

Figure 3: Preview of the original solution of the task

The editor, that is in fact content editable div tag, is also adapted for the programming language LOGO. It supports syntax highlighting so the words that are recognized as a part of the LOGO syntax are represented with blue color. Autocompletion of commands is also supported in the editor, so that when user types in the editor he gets a list of LOGO commands that are adequate for the input string. When the user clicks on a command from the suggested options on the list, a tooltip is shown next to the clicked option. This tooltip contains an explanation for the correct usage of the command and its meaning in LOGO language (Fig. 4).

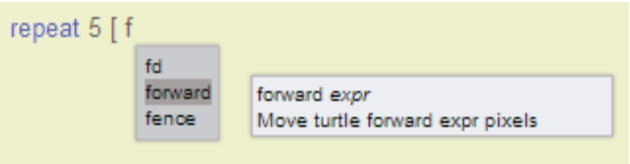

Figure 4: LOGO editor

Using this feature, the student, even if he was not familiar with some command before, gets the opportunity to learn its usage, so he can use it when needed. If the student double clicks the option from the list of offered commands, the command is automatically inserted in the editor without the need to type the whole word. As mentioned earlier, the editor is a div tag that has the attribute contenteditable set to true, so the content of the element can be changed. The div with editable content is transparent and it is overlaid by another div which contains the syntax highlighting.

The table containing the explanation of the commands is set on the right side of the page. The commands in the table are grouped in six sections and are ordered by their importance and frequency of use. These groups are: turtle graphics; flow control; numeric operations; lists; text and procedure definition. The commands that were implemented and available in the application, and the way they are grouped is according to the Terrapin LOGO implementation [7]. There are many more LOGO commands, but in this implementation some of the most fundamental ones are used.

The user can try to solve each succeeding task using the commands that he/she has learned in the previous tasks. Then he can compare his solution with the original one. If a new command is used in the original solution that the user was not familiar with before, he can see the explanation of that command in the language reference table and try to solve the same task using that approach. Even if he/she makes a mistake, an appropriate error message will appear to notify him/her that he/she has made a mistake and thus the shape will not be drawn. The error message will contain the command where the mistake was detected, followed by an explanation of how to use the command correctly. This decreases the probability of making the same mistake in the next attempt since an explanation of how to do it appropriately has been shown earlier. The child can try to draw the same shape using many different ways and trying different combination of commands. In this way the student gets to know the insight of the LOGO commands without the help of a tutor.

We can say that the application promotes interactive learning in some way, because with the auto completion feature of the editor the user can learn new commands while he practices and improves those that he already knows. For example if the child writes the repeat command he/she gets all the commands that start with the letter R. So he/she can become interested in the new commands he/she discovered, learn their meaning and try to use them in that or one of the succeeding tasks.

The student can see all his previous attempts in the history panel as shown on Fig. 5, where he/she can have a look at the commands used in his previous tasks. At the same time the student can have an insight of the progress he/she had made in the learning process.

The application is simple and inexpensive for maintaining since it does not require many resources. Almost everything is executed client side, so the load of the server is very light.

The only request to the server is made when the page loads to get the predefined tasks from the database. Then the tasks are set in a list, and the navigation through them is also client side. Because of this, the application runs fast and the user gets immediate response when executing the input commands. In comparison to the user page, the admin page is not completely client side, except for the execution of the commands. The predefined tasks are stored in a server side database, so the manipulation with the tasks is server side. Considering the fact that it is a matter of very few and simple commands that are not created and changed that often, this activity does not produce any real load for the server.

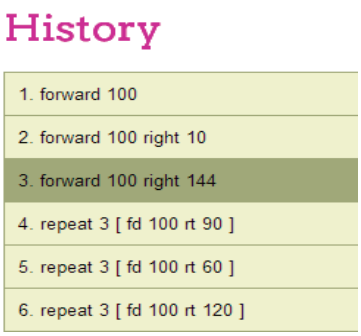

Figure 5: History panel with previous attempts

IV. CONCLUSION

The LOGO programming language has demonstrated to be very effective when it comes to education purposes especially for lower grade students and children. It started as a method for learning mathematics, but the benefits are more profound. It is very useful for learning geometry, algebra and learning the main logic and concepts in programming.

The goal of this implementation of LOGO language was to demonstrate the process of learning by writing LOGO commands in a code editor in order to solve certain predefined tasks. The code editor is adapted for LOGO programming language and supports syntax highlighting, autocompletion of LOGO commands and brief description of commands in a form of a tooltip. The application also contains table with explanation of LOGO commands in the user page. The commands in the table are grouped in six sections by the functionality and the order by which they have to be learned and used in the tasks. When user executes the input commands from the editor, if they were correct, he would get immediate visual output. In a case of an error, the user would get message containing an explanation about the correct usage of the command in order to avoid making the same mistake again. Even if he does not know how to solve the task he can see the original solution and retry to solve the task using that approach.

The application is implemented as web application using HTML5 canvas element for the visual output of executed commands and JavaScript for the code editor and the interpretation of LOGO commands. Thus, the whole processing is client side, which means it does not overload the server with requests on every action.

The tasks given to the user are defined and managed by the administrator. They are organized systematically by increasing complexity and by the order that the commands should be learned, according to the language reference table. Each succeeding task is more complex and demands more effort and reasoning from the user than the previous one. Each next shape consists of the previous ones, i.e. the commands of the previous are a subset of the commands used for the next one, so the child would have to understand and solve the previous tasks in order to be able to resolve subsequent tasks. At the beginning when the student starts to use the application he knows only few or maybe even none of the commands, but by using the reference table with explanation of the commands and by looking at the solutions of the given task he starts to learn the commands and their meaning. By practicing the commands he discovers the many different approaches for solving the problems, without the help of a mentor who will lead him along the way.

REFERENCES

[1] MIT Logo Foundation web page, http://el.media.mit.edu/logo-foundation/logo/index.html accessed 01.03.2013

[2] Pea, R. D., LOGO programming and problem-solving, in *Chameleon in the classroom: Developing roles for computers*, Symposium presented at the annual meeting of the American Educational Research Association, Montreal, 1983.

[3] Weyer S. A. and Cannara A. B., Children learning computer programming: Experiments with languages, Curricula and programmable devices, (Technical Report No. 250) Stanford Univ., CA. Inst. For Mathematical Studies in Social Science, 1975.

[4] Mendelsohn P., Green T.R.G. and Brna P., Programming languages in education: the search for an easy start, in T. Green; J.M. Hoc, R. Samurcay, D. Gilmore. *Psychology of programming,* Academic Press, New York, 1990.

[5] Geva E. and Cohen R., Understanding turn commands in Logo: A cognitive perspective, *Instructional Science*, Volume 16, Issue 4, pp. 337-350, Kluwer Academic Publishers, 1987.

[6] Structure and implementation of computer programs, http://wla.berkeley.edu/~cs61a/fa11/61a-python/content/section_notes/week12/week12.pdf accessed 01.03.2012

[7] Terrapin tool for thinking, http://www.terrapinlogo.com/indexhelp.php accessed 03.03.2013

[8] Abelson H., Goodman N. and Rudolph L.,. *Logo Manual*. Artificial Intelligence Memoranda at Institute of Technology in Massachusetts, 1974.