Puzzle-like programming languages in teaching programming

Vedran Juričić and Matea Radošević
Faculty of Humanities and Social Sciences, Zagreb, Croatia
vedran.juricic@ffzg.hr, matea.radosевич@ffzg.hr

Abstract - This paper examines use of puzzle-like programming languages in teaching programming. It analyzes a variety of puzzle-like programming languages and their availability in Croatian elementary and high schools and textbooks. Also, it compares available puzzle-like programming languages by criteria that should be satisfied to be recognized as programming language that is suitable for teaching programming.

Keywords – teaching; programming; puzzle-like

I. INTRODUCTION
Computers play a large role in our everyday lives and familiarity with computers and programming is becoming essential. Methods to introduce computer programming to children are being explored by researchers and teachers. It is shown that learning how to program computers can have a positive effect on children’s achievement in math and science, but also in language skills, creativity and social-emotional interaction [1].

Learning to program is similar to learning a new spoken language. The content of a first programming course should focus upon syntactic and mechanical learning to certain extent. Without knowing syntactic rules of a language, programs cannot be written. Mathematics and foreign languages have traditionally been taught through practice and drill. Repetitive drill learning can be appropriate technique for teaching syntactic rules in a first programming course [2]. But drill way of teaching is boring for students and it is difficult to motivate them. It is important for students to be motivated intrinsically when learning so they can achieve better results that way.

Nowadays, there are at least three different types of programming languages: image or picture languages, block or bubble or puzzle-like languages and “real” programming languages [3]. Image programming languages use images and other visual methods to bring the basic ideas of programming closer to students. Block programming languages use blocks with parts of code written on them to solve a problem. Block programming languages represent a transition from image programming languages to traditional programming languages. The “real” (traditional) programming languages are actually textual programming languages but, if explained step by step, they can also be used as teaching programming languages.

In this paper we describe puzzle-like programming languages used to teach children hot to code. Puzzle-like programming languages look like a set of drag-and-drop code fragments.

II. PUZZLE-LIKE PROGRAMMING LANGUAGES
Puzzle-like programming languages (visual programming languages, block-based programming languages) are programming languages that let users create programs by manipulating with elements graphically rather than by writing textual commands. These programming languages may be classified according to the type and extent of visual expression used, into icon-based languages, form-based languages, and diagram languages. They provide graphical or iconic elements which can be manipulated by users in an interactive way according to some specific spatial grammar for program construction.

The general goal of puzzle-like programming languages is to make programming more accessible to novice, mostly children, and to support programmers at three levels: syntax, semantics and pragmatics [4].

A. Popular puzzle-like programming languages
Puzzle-like programming languages are becoming more and more popular in education. They differ in their characteristics, features and possibilities. This paper analyses the characteristics and features of five most popular of puzzle-like programming languages: Alice, Kodu, mBlock, Scratch and MakeCode. Those languages are then compared following 17 criteria and checked if they meet the required criteria.

B. Criteria
Creators of programming languages for teaching programming, Saymour Papert (creator of Logo), Niklaus Wirth (creator of Pascal), Guido can Rossum (creator of Python) and Bertrand Meyer (creator of Eiffel) suggested criteria that programming languages for teaching programming should satisfy. [5] There are 17 criteria that are grouped into related sections.

Learning:
- The language is suitable for teaching. That means that the language should have simple syntax and natural semantics, and abbreviations should be avoided.
- The language can be used to apply physical analogies. To meet this criterion, language should
provide multimedia capabilities without extension.

- The language offers a general framework. In other words, the language should let beginners learn fundamentals that will serve as basis for learning other programming languages.
- The language promotes a new approach for teaching software. That means language should cover many aspects of software development process, not only implementations. The language for teaching programming should be designed to cover entire methodology for constructing software, both language and principles, tools and libraries.

Design and environment:

- The language is interactive and facilitates rapid code development. Environment that support use of the language should allow beginners to implement new ideas. Also, it should give students feedback on their progress in an interactive way.
- The language promotes writing correct programs. It is important to avoid “trial and error” approach to writing code. Students should be taught to write correct code without bugs.
- The language allows problems to be solved in “bite-sized chunks”. In other words, it should allow students to focus on one part of the problem without moving on the next one. The language should support problem decomposition using procedures, functions etc.
- The language provides a seamless development environment. The code writing environment should have an intuitive GUI. It should make features and libraries easily accessible for beginners and for advanced programmers.

Support and availability:

- The language has a supportive user community. There should be support for all users, students and everybody interested in learning that programming language. Support can be in form of course books, tutorials, web pages, documentation or some other form.
- The language is open source, so anyone can contribute to its development. The language should be product of group of people that does not want to profit from the creation of the language but let anyone be part of the process of making a programming language suitable for teaching programming.
- The language is consistently supported across environments. It should be accessible from different platforms.
- The language is freely and easily available. It should be available worldwide, and should not be charged for using.

- The language is supported with good teaching material. That means there should be textbooks and other teaching/learning materials that follow the selected programming language.

Beyond introductory programming:

- The language is not only used in education. It should be suitable for developing real world applications. In other words, it should be applicable in various areas, not only in education.
- The language is extensible. It should let programmers access advanced functions.
- The language is reliable and efficient. It should be useful also in creating bigger applications. It is not very important from pedagogical perspective but programming instructors feel it is an important criterion.
- The language is not an example of the QWERTY phenomena. It means the language should be useful nowadays but it should also work well in teaching programming fundamentals that do not change in time.

III. PUZZLE-LIKE LANGUAGES COMPARISON

In this chapter, five programming languages will be described and compared. There will also be shown how does the programming code for calculating the sum of the first ten even integers look like.

A. Alice

“Alice is used by teachers at all levels from middle schools (and sometimes even younger) to universities, in school classrooms and in after school and out of school programming, and in subjects ranging from visual arts and language arts to the fundamentals of programming and introduction to java courses [6].” It is a programming language used mostly for teaching programming with offers students a full transition to the Java programming language. There are four versions of Alice and one should choose a version according to their knowledge and interests.

Alice programming language is definitely a language suitable for teaching and specially teaching younger students, although it is said that Alice is used as teaching language also at universities. It is applicable between operating systems. Alice offers a general framework and helps teach software in new, modern way.

Also, Alice is an example of interactive programming languages that helps students learn how to write correct codes, although it sometimes relies on “trial and error” principle, which is often the way younger students learn everything, so it is also a way of learning how to program for the youngest students. The language has an intuitive GUI, but it does not allow programmers to solve problems by cropping the problem onto smaller parts.

Alice has sufficient support for students and teachers, it has many teaching materials on their web-site and it also has forum that is often updated. The language is partly open source, but it is available under different platforms.
It is free and easily available worldwide. Unfortunately, it is not available on many languages, so it is mostly downloaded in English.

Alice, as a programming language, is used mostly in education. It has a Java in background so student can see how would the code they wrote look like if they wrote it in Java programming language. That is why Alice represents a good transitional programming language in teaching programming. Alice will help students understand the concept of programming. On the other hand, it is not useful in creating high speed applications nor is it helpful in writing more complicated programming codes.

B. Kodu

“Kodu lets kids create games on the PC and Xbox via a simple visual programming language. Kodu can be used to teach creativity, problem solving, storytelling, as well as programming [7].” It is a programming language that is created for teaching youngest students because of environment and features. It is suitable for teaching and it represents a new way of teaching programming. On the other hand, it is not applicable in different operating systems and does not offer a general framework because it, almost in general, relies on creating games for the youngest students.

Kodu is interactive programming language and encourages students to split the problems they need to solve in smaller parts but it mostly does work as an “trial and error” way of writing codes. The GUI is designed for basic programming. There is support for both students and teachers, there are even competitions that motivate students to write their first codes. It is available under different platforms and is free from subscription.

Although students can create games in Kodu, those are simple games. That is why Kodu is used only in education. It helps beginners understand the basic concepts of programming but it not useful for solving some more complicated problems.

C. Scratch

“Scratch is a programming software tool that is designed by MIT and is one of the most popular programming languages for children. It makes programming as easy as building blocks [8].” Scratch is a real example of puzzle-like programming languages. Parts of code are written on blocks that are shaped and colored differently, but intuitively demonstrate beginners which part fits next to each other.

Scratch is suitable for teaching and works across different environments. The language offers a general framework and is a good example of the modern approach for teaching software. It is interactive and promotes writing correct programs, but still mostly relies on “trial and error” method. Fig. 1 shows how does the programming code that calculates the sum of the first ten integers look like.

Scratch supports modularization and has an intuitive GUI for both design and implementation. It has a very supportive user community with lots of teaching materials and tutorials, forum etc. It is open source and supported across environments.

There are different versions of Scratch in different languages. Also, Scratch gives a possibility to write code online. It is used only in education but in the background of Scratch there is C++ programming language. In that way students can see how their code would look like written in C++. It helps students to transit from puzzle-like to textual programming languages. Scratch gives students good introduction in programming and helps them learn how to think logically.

D. mBlock

“mBlock is a STEAM programming software tool designed for programming for kids. It is developed based on Scratch and Arduino code. Fig. 2 shows programming code that calculates the sum of the first ten integers. It can be seen that the code looks almost the same as the code written in Scratch programming language. It supports block-based and text-based programming languages. With mBlock 5, children can not only create games and animations by dragging blocks or using Python code, but can also code robots or boards to do anything they can imagine [9].” mBlock is a programming language that meets the same criteria as Scratch but also represents a
programming language that is used in programming physical objects. That is why it demonstrates students what is the real effect of their code.

E. MakeCode

“The MakeCode editor provided by Microsoft makes it easy to program your micro:bit with blocks and JavaScript [10].” MakeCode editor is, just like Scratch and mBlock, a real example of puzzle-like editor. Fig. 3 shows the code for the sum of the first ten integers. It looks similar to code written in Scratch and mBlock, but it the parts of the puzzle needed to build the code has some tiny differences.

![Figure 3. Code written in MakeCode programming language](image)

It is used mostly for educational purposes. It helps students understand the basic concepts of programming and algorithmic way of thinking. It is a good way for students to go from puzzle-like to textual programming languages because MakeCode editor gives them opportunity to see how the code they wrote looks in Python. As MakeCode is used in writing code for micro:bit, it also allows students to get the real feel of what their code does as it effects on what happens with their micro:bit.

F. Additional comparison

There are some other features that can be observed when evaluating programming languages for teaching. The features of programming languages analyzed in this paper are listed in Table 1. That shows a year when each programming language was first and last released (First version, Last version), operating systems they can be installed on (Operating systems), how many users worldwide do they have (Users) and what kind of user support (Support). It can be seen that Alice was released 20 years ago, while mBlock and Make Code are relatively new. Scratch was publicly launched in 2007., but its first prototype was made in 2002. All programming languages have their last release in 2018, or even in 2019., and that proves their active community and development.

<table>
<thead>
<tr>
<th>Table 1: Additional Comparison of Programming Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>OS</td>
</tr>
<tr>
<td>Users</td>
</tr>
<tr>
<td>Approx. age</td>
</tr>
<tr>
<td>Exp. level</td>
</tr>
<tr>
<td>Physical objects</td>
</tr>
</tbody>
</table>

All languages can be installed on Windows operating system, and all languages except Kodu support macOS and Linux operating systems. mBlock, MakeCode and Scratch have well-built community support. Forums and websites give relevant information, and teaching materials are available and free. There are also events and competitions that specially motivate students.

It is shown for which age groups these programming languages are suggested for (Approximate age). Most languages are suitable for teaching youngest students. Kodu, mBlock and Scratch are suitable for teaching children who did not even enroll in elementary school.

Table 1 also shows how much experience students should have to start programming in each programming language (Experience level). As the languages are suitable for teaching the youngest students, they do not need to have any experience. It is desirable that students have little or some experience in programming when writing code in MakeCode editor.

mBlock and MakeCode programming languages are suitable even for programming physical objects, for example, simple robots (Physical objects). That gives students better understanding of algorithms and basic concepts of programming. The possibility of programming physical objects is one of the biggest advantages of mBlock and MakeCode.

Kodu gives students opportunity to create their own games and that makes it easier for teacher to motivate students to start writing their first codes. That is the biggest advantage of Kodu programming language. The biggest advantage of Alice is existence of many official textbooks for teaching programming using Alice. Both Alice and Kodu do not have as good support as MBlook, Scratch and MakeCode have. On the other hand, mBlock, Scratch, Kodu and MakeCode are not that present in schools all over the world as Alice is.
G. Puzzle-like programming languages in Croatian schools

In schools in Croatia there are few programming languages used to teach programming: Logo, Python, QBASIC. While taking part in programming competitions, students also use Basic, C, C++ and Pascal. In some schools, students write their first programs in Scratch, MakeCode and mBlock programming language [11].

Puzzle-like programming languages are not common in Croatian education. They are used for teaching programming as extracurricular activity, but not in schools. Some teachers in elementary schools teach their students to write code in Scratch, MakeCode or mBlock. There are no official teaching materials (textbooks or digital materials). Teachers can find materials on official websites or education forums. Also, there are plenty of examples that are good as motivation for beginners. The main problem is language. Most of the materials are in English so teachers need to be fluent in English and able to translate all the materials in Croatian.

IV. CONCLUSION

As computers and programming are becoming par of everyday life, children are taught to write their first codes at early age. For youngest students, writing code in traditional textual programming languages can be a problem because of lack of basic knowledge. Puzzle-like programming languages let children start programming even before enrolling elementary school. Most popular puzzle-like programming languages meet the criteria written by creators of programming languages for teaching programming. Some of the puzzle-like programming languages are suitable only for educational purposes but they help teachers motivate students and teach them basics. MakeCode programming language could be the most suitable for teaching programming in elementary school as it has simple syntax and students can see the output on the little screen. For secondary school students, mBlock is great programming language for teaching since more complex codes can be written and they can also see the product of their work by the moves of the physical object.

In Croatian schools, puzzle-like programming languages are not used very often and there are almost no teaching materials. However, some teachers manage to use puzzle-like programming languages in classrooms and in that way introduce programming to their students in modern way.

REFERENCES