CodeMage: Educational Programming Environment For Beginners

S. J. Whittall[1], W. A. C. Prashandi[2], G. L. S. Himasha[3], D. I. De Silva[4], T. K. Suriyawansa[5]
Faculty of Computing
Department of Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
[1]stephano.whittall@gmail.com,[2]chathuminiwij@gmail.com, [3]himashaliyanage@gmail.com
[4]dilshan.i@sliit.lk, [5]kushnara.s@sliit.lk

***Abstract*— CodeMage is an interactive educational programming environment targeted at novice Java programmers who have little knowledge in basic programming. The system comes with innovative features such as, real-time guidance and reviews, code generation tool, visual debugger, hinting system for best practices, remote tutor and database manipulation tool which makes it a unique educational programming environment. Though many of programming tools are available in the market, they have their drawbacks in suitability to be adopted in programming by beginners due to the complexity in software interface, cryptic error messages and warning messages, no adequate support to fix errors and not adhering to real-world programming context etc. CodeMage is an attempt to overcome above problems and change how novices perceive and practice Java programming around the world and it is working as expected.**

Keywords—Educational programming environment, novice programmers

I. INTRODUCTION

Our reliance on technology will only increase day by day. The young generation of today must be able to not only consume this technology but also to understand and control it. The ability to write programs is the ability to create change. Along with this huge digital shift, programming is known as the new literacy in today's world [1].

Exposure to a firm and interactive educational programming environment for novice programmers at early stages may make it easier to grasp advanced concepts and procedures at later stages. According to recent researches, novice programmers endure a wide range of difficulties and misconceptions for decades [2]. Introductory programming tools are generally labeled as difficult to follow and often have the highest dropout rates.

It is been found that beginners find it difficult to adopt from those introductory tools specific languages to common programming languages. According to the conclusions of surveys, Java is one of the widely used programming languages [3]. Taking this into consideration. So, one of our goals is to expose beginners to the real programming environment, but with step by step guidance and continuous encouragement towards the subject in a user-friendly manner.

Most of the programming introductory tools out there are based on visual programming languages. These tools direct cognitive minds of beginners to interact with visualization objects rather than trying to make them understand the programming concepts hidden behind. There is a chance of them implanting erroneous impressions in young minds about the subject. Exposing novice programmers will result in the need to unlearn what they have already learned and it will be a difficult task [4].

A survey carried out at Monash University indicated that "Programming was considered the most difficult and least interesting subject by most first-year students in all Computing courses" [5]. Some of the core problems which novice programmers encounter are difficulty in grasping the conceptual value of elements, unavailability of feedback for the particular programming task, fear and lacking of confidence to experience things by their own. Therefore a continuous guidance and feedback should be given to novice programmers.

Error messages are one key factor of an educational programming environment as it allows beginners to identify and learn how to overcome mistakes and misconceptions. Students have trouble in typing syntax properly and need a way to type them without much effort but still learning and remembering the syntax without any misinterpretations.

This paper initially introduces to CodeMage and the related existing work is mentioned in the next section. The readers' attention is then drawn to research methodology. Section IV provides a discussion about the research work and section V concludes the paper.

II. LITERATURE REVIEW

Novice programmers, as well as experienced programmers, need to interact with environments which deliver access to tools which they must use to accomplish their programming tasks. More experienced programmers are used to work within more sophisticated and advanced Integrated Development environments which provide additional capabilities and functionalities which are advantageous to achieve their programming goals. However, when using IDEs designed for experienced programmers, novice programmers consume excessive time to overcome the complexity of the tool and create a spurious fear in their minds about programming [3].

978-1-4673-9077-4/17/$31.00 ©2017 IEEE

To address this myriad integrated development tools have been designed for novice programmers.

Traditional introductory programming environments composed of text editor, compiler, debugger and a runtime environment for program creation, compilation, debugging and simulation, execution respectively. Existing IDEs designed for the novice such as BlueJ [6], DrJava [7], jGRASP [8] are a simplified version of advanced IDEs. Those IDEs identify and highlight syntax and runtime errors done by users. Compared to experienced programmers, cognitive minds of novice programmers' shows weak performance in mapping logical solutions and tend to implement wrong logics. Unlike existing Integrated Development Environments CodeMage eliminate misinterpretations and guide to achieve goals by identifying and highlighting logical errors. The identifying, highlighting and guidance are given to fix all syntax, runtime, and logical errors makes CodeMage more unique.

Recent researches conducted by observing a set of undergraduate novice programmers indicated that novice programmers make lots and lots of errors, and they spend a considerable amount of time struggling to fix those errors and most of them couldn't even execute the written code [error messages]. One of the main reasons for this is that error messages displayed from traditional IDEs are complex and difficult to understand even for experienced programmers. As a remedy, some of existing introductory programming tools avoid the occurrence of errors using various approaches. Blockly [9] avoid users from making syntax errors by using the block based approach. Scratch [10] avoid syntax errors by using visual based programming approach. But these implementations drags the standard programming experience away from novice programmers and implant the impression of "nothing can go wrong" in minds of novice programmers. This results in damaging the mentality of novice programmers when they start to deal with complex error messages and advanced programming tools. As the solution, CodeMage converts complex and lengthy error messages into more user-friendly and easily understandable message.

A visual programming introductory tool is a tool which allows novices to create programs by manipulating graphical program elements rather than specifying them textually. Existing introductory programming tools such as Alice [11], Scratch [10] and etc. have taken an approach to implant basic programming concepts by allowing novice programmers to interact with virtual environments. Alice is a 3D programming environment designed to teach object-oriented programming in the context of creating animated movies and simple video games interacting with a virtual environment. Scratch is very similar to Alice. The main concern of scratch is to provide a multimedia environment that allows the students "to put together fragments of computer programs, try them out, take them apart, and recombine them" [12]. Students are allowed to create storyboards and simple video games by combining graphical code blocks together. Humans are good at processing graphical information, but on the other hand, most programming concepts and data structures are dynamic abstract concepts with no graphical form. Adapting graphical elements in teaching programming concepts for novices directs their attentiveness to interaction with graphical elements rather than grabbing the programming concepts hidden behind it. Programs and algorithms are dynamic artifacts. So apprehending their essentials is a challenging task for novices. Students may keep their eye on dynamic visualizations without compassionating the deep concept behind it. According to researchers cultural aspects, individual differences, and ambiguous objects and effects are causing difficulties [13].Moreover, visual based programming tools, specifically in the context of exposing novices to programming, have scalability concerns which make it difficult to adapt to a larger environment and the lack of standard rules in visual styles results in quality issues of a tool [14].

Block based programming environments are a kind of visual programming languages in which learners can assemble puzzle pieces together by mouse drag and drop. The programming environment gives the learner visual and sometimes audio feedback to inform the user about the validity of plugging blocks with one another. They use their own natural language to label the blocks to communicate their function. The blocks are categorized according to the functionality and the execution is also done visually. One critical drawback of using blocks is that novices assume "everything has its place" which is a myth in programming as commands can be embedded in various places within a program to produce a number of behaviors. Research has found that in most of the textual language tools learners will have to perceive errors as wrong rather than errors as fixable due to the way they display the errors and the restrictions and less support is given to correct syntax errors [15]. The users that have used Snap! , a block based tools claim that the tool is less powerful although they know a little about programming they have found that the blocks are limited and there is much larger and complex stuff they can do with java. The research also shows that the users of Snap! Complained that it consumes a lot of time to compose a program by snapping the mouse. "If you want a specific block and it's not there, you're going to have to put a lot of blocks together to make it do what you want it to do" is the idea that the novice programmer gained. Furthermore, there were comments by students such as, "Java is actual code, while Snap! is something nobody will let you code in." This same point was made by another student who said: "if we actually want to program something, we wouldn't have blocks. This shows that beginners programming tools have to adhere to advanced programming contexts. This drawback seems to affect older learners who are willing to develop programming skills [15].

Drag and drop blocks based languages such as Blockly [9] and GameMaker [16] are found to be more fun by novice programmers as the chance for them to make mistakes is reduced by that approach. But on the other hand, text-based languages are considered as the language that programming majors use and more valuable in education as it enables the learner to go in depths to learn and write complex pieces of code. The configuration in Alice and Scratch is allowing students to drag and drop code into code editor to create a story. These tools focus on the logic of piecing together a program rather than the syntax [11, 10]. Allowing novices to drag and drop code to a code editor at the first step of exposing to the programming environment is plummeting their strength of committing essential keywords to memory. Visual programming environment Scratch has been evaluated against textual programming environments like Logo by research [17]. When using scratch the blocks user drags and drops to the editing space are locked into one another if only those blocks are valid syntactically to be next to each other. So typing errors and misremembered language details has no chance of occurring. But semantic errors will occur. As Logo is a textual representation novice will have to focus on the syntax errors.

As the main goal of a novice programming environment unremittingly engages with cognitive minds of future programming generation, best programming practices must be

encouraged and induced from the very beginning. According to recent researches, poor use of coding standards can often lead to software failures and cancellations. Encouraging proper programming practices and emphasizing the importance and benefits of using them at the very first step can minimize the software failures and cancellations that may cause in the future. Currently, available tools such as BlueJ, Scratch, Alice, and Greenfoot [18] occasionally implant some wrong ideas about simple keywords and concepts in programming. Moreover, they encourage some common mistakes done by programmers. As an example in Scratch, array indexing starts with one. But in almost all the programming languages array indexing starts with zero. Using one as the first index is a common mistake done by programmers [19]. Encouraging this practice is cultivating wrong programming habits among novice programmers. And another common bad convention practiced by Scratch is using variable assignment operator as the Boolean comparison operator. These common bad practices increase the gap between introductory programming languages like Scratch and industry level programming. According to the theory of unlearning it is hard to unlearn stuff more than learning. So if introductory programming tools implant wrong ideas about programming in young minds for a significant period of time it makes it hard for novices to minimize common mistakes of programming. Method-invoking is a basic concept in object oriented programming. But in Greenfoot this concept has been totally ignored and it may implant a wrong idea about method invoking.

Myriads of individuals connect to the web day by day. Web-based software tools are tools which are used over the internet with a web browser. Most of the novice programming tools such as Alice, Scratch, Greenfoot, BlueJ and etc. are desktop applications. These tools need to be downloaded and updated time to time in order to get latest features. Unlike web based applications desktop applications does not support portability. Relevant information cannot be accessed at any time from any computer as in web based applications.

Some of the existing programming environments utilize narrators to give audio or visual feedback for novice programmers to support them. Of them some are aimed at lowering the demotivation experience of novice programmers while giving them feedback. Such tools include designing languages that imitate how children describe the behavior of the program [20]. Some other approaches have been to simplify debugging by allowing the users to ask “why” questions about the programming output [21]. The issue with this kind of tools is that they are not user-friendly enough for users to understand the errors and it might even take time unnecessarily which would be annoying. Tools like Greenfoot [18] shows the hard to figure out errors from Java compiler itself which novices would take as a sign of personal failure. Another tool named Gidget [22] was designed in a way that it tasks users with helping a damaged robot by debugging its damaged code. They have personified the system so that it conveys information about errors. But as Gidget is characterized in a way that it shows emotions learner might get distracted by the face rather than being attentive to the personified text. In the existing programming introductory tools, the narrator does not represent errors in the verbal format. Instead, most of the narrators use sounds to output error. But in the CodeMage context, it will output errors in the verbal format through the narrator working on top of an underlying text to speech mechanism. Moreover, narrator is acting as an assistive tool which delivers continuous guide and support to the novice programmer. If the beginner can't read that errors at that point? Just get the help from the narrator read it to you. CodeMage researched through the Chrome web store to find the best free text-to-speech Chrome extensions and came up with some good TTS tools. When the student spent its entire day at work looking at a screen, reading the news or a feature on yet another screen isn't the best way to recuperate. Which is why CodeMage is coming with a real time narrator. But as it turns out, you can go one step further and take your eyes completely out of the textual reading.

Very few of existing systems like code.org [23] provide the user with animated demonstrations to show the behavior of the code. But researchers [21], have noted that for loops are confusing to novices as they fail to understand the behind the scenes of it such as loop count being updated. And also various studies done on the novices conclude that they try to view the meaning of the code line by line rather than looking at it as chunks of meaningful lines of code. Simulation is a very effective technique in teaching. When a program which is hard to understand is simulated using some sort of a technique, it helps the observer to capture the whole idea of the scenario. Similarly, in programming, there are many places where the programmer requires to understand each snapshot of the execution instance. But once the program is compiled and run the whole output appears in seconds. In that case, if a programmer required to do some debugging, it may not be helping the programmer to find if there are any erroneous outputs during the execution. But rather than appearing the whole output as a whole, if the output could be seen step by step, then, the programmer could easily fix the code which causes erroneous outputs. In this research, it has addressed the above-mentioned issue and has provided a successful solution. Once the code is typed in the IDE and compiled, the programmer has the capability to look into all the variable changes and all the loop structure behavior in the code segment. Step by step execution is simulated in each line of execution. This research product deals with the use of interactive simulation and animation of code snippets. This could provide students with a motivating introduction to the java coding. This could help the potential users to Integrate fundamental skills and principles with the high coding attempts. Moreover, it encourages the novice learners to learn the code quickly and conveniently.

III. Methodology

The goal of our study was to investigate the role of our solutions suggested to address the issues we identified as faced by novice programmers who had little programming experience when using programming environments. For this, we designed CodeMage with several innovative features and allowed a set of 83 students to use it under the supervision of authors of this paper. After that, we carried out a survey to measure the usefulness of each and every feature. CodeMage is a web application that consists of a 3 tier architecture making it a distributed system according to its functionalities to guarantee its scalability and availability. The presentation tier is developed using AnguarJS version 1 and HTML5. The backend is implemented using JavaEE 7 Technology using Spring MVC 4. The rest of this section will give a brief introduction to a few of the features available.

A. Editor

The editor is where the user can perform all coding tasks. The editor has self-driven driven functionalities which will perform syntax highlighting and error highlighting. Code indenting & code suggestion is also supported by the editor. Using a predefined set of rules the possibility of user’s code to

be semantically and logically erroneous will be detected while the user is typing the code.

B. User-friendly error messages & error correction guidance

A stack trace would simply provide you with information such as class, line number, and error. CodeMage will provide further information than what is displayed on a stack trace to expose the underlying causes that the error message may not convey. For an example, a "NullPointerException" could be caused by invoking a method on a null object, accessing or modifying a null object field, taking length from a null array, etc but the stacktrace will only give you the name of the error. CodeMage will give you the exact cause and the root position of what may have caused the error. The default java error message will also be converted to a more meaningful error message by analyzing the source code and by extracting information from java stack trace. This is implemented by analyzing the user's source codes with the help of Java parser and abstract syntax tree which will generate a modified version of user's source file which in turn will be run using Java reflection to extract useful information to generate user friendly error messages using algorithms. Refer figure 1 and figure 2.

CodeMage's error message will consist of several parts such as a way for users to find where and why the mistake occurred, explanation to assist with the correction and contextual information about the error.

C. Visual Debugger

After having a code that would compile successfully but which may have runtime errors it would be analyzed to troubleshoot the issue. It will allow users to watch the execution of the program and give a visual simulation of the variables while executing line by line of the program. CodeMage will have highlighted the source code while simulating the value change of the visually displayed variables. Additionally, the above simulation blocks will be placed inside scope blocks to give a better visual aid of the local scope. Reverse execution is also possible.

D. Hint System

The code will run through a static code analysis to determine style errors and dodgy code and inform the user to inculcate best programming practices in them from an early stage. This functionality is done by using Findbugs tool on the server side then restated to the user in a neat manner.

E. Narrator

The narrator is assigned with the task to interact with the user and communicate messages between CodeMage and the user. The narrator has two medians of communication both voice and textual. The user interface has a small space on the bottom right of the screen where the narrator resides. Using chat bubbles the message is represented to the user. The narrator will guide the user on how to use the tool, convey messages on contextual information about errors, give examples of common causes of errors and fix syntax errors for the user giving him an immense support to make sure the user does not feel alone and helpless and keep the user motivated.

> The program encountered an error in file Hello.Java at line 21 in class, Hello inside method print().In Java, this error is called as NullPointerException.It seems that the code tried to call method CharAt from an object str which is null.Make sure str has been initialized in the program. You may need to initialize the object using the keyword "new".

Figure 1: CodeMage runtime error message

F. Code Generator

The code generator works as an aiding tool to reducing keyboard strokes while coding. Simply the user select the control structure that is needed and fills in the values and the control structure is generated on the editor. The same function could be done to wrap a certain set of codes within a control structure. In order to achieve this, the user first has to select the code segment that needs to be embedded then select the control structure and it will wrap itself around the selected code segment.

Another useful functionality that the code generator provides is that it gives the opportunity for novices to use Java inbuilt functions without even knowing what function they should use. For example, if the user wants to split a string using a delimiter he/she has to go through functions available and choose the function that describes their need in natural language. The code generator will generate the suitable function.

G. Remote Tutor

The remote tutor acts as a real-time sync agent between two clients. One person using the editor to write a program can share that with multiple people in real-time which will provide the people watching with the ability to change the synced code block or even execute it as their own copy. A simple remote share session can be initialized by clicking share editor. Which can then be subscribed by another listening party who will receive real-time updates of the shared editor the party watching can do changes and run the code that is been watch from the broadcaster from his or her own instances not causing any changes to the broadcaster.

H. Database Manipulation tool

CodeMage database manipulation tool will be doing some major functionalities to make self-learning task easy and effective. The main functionality is to generate SQL queries and Java codes for database functions. Users are allowed to perform DDL, DML, DQL based database functions in few button clicks. One would create a database by simply entering the database name and the query will be generated in real time. Within the process users are allowed to create, insert and delete tables for selected databases and the both SQL query and java code is generated by the tool. Query builder is another main section where users are facilitated to build SQL queries simply and effectively.

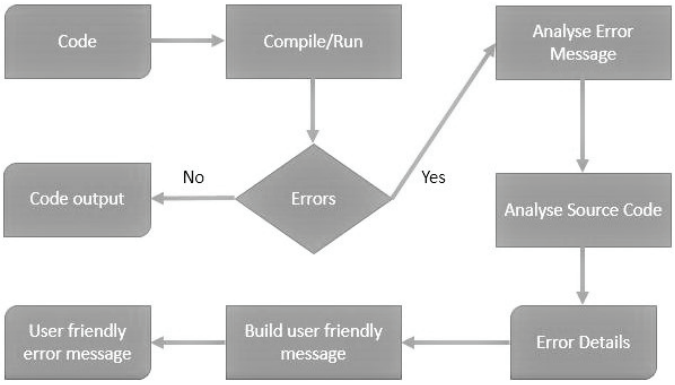

Figure 2: Conversion of cryptic error message to user-friendly error message

IV. Discussion

In this chapter, we sought to derive conclusions by comparing the user experience of CodeMage with other programming environments used by novices. We randomly selected 100 undergraduate IT students and conducted a survey to get a measurement of severity of known problems which novice programmers face. The results validated all our findings in the literature.

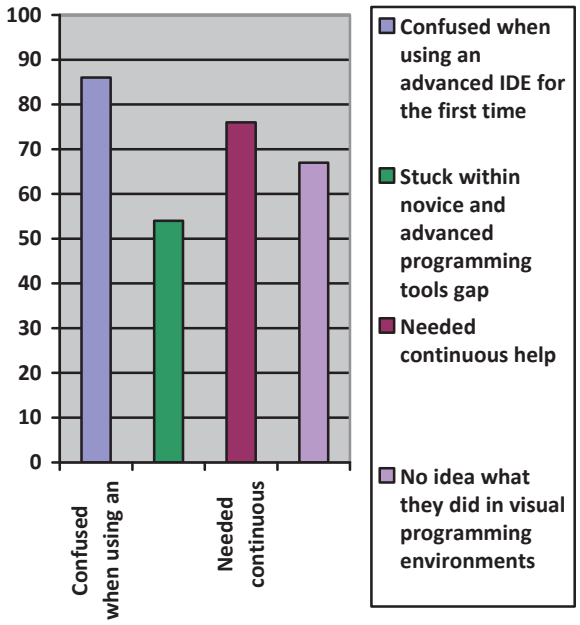

Figure 3: Results of the survey conducted before introducing CodeMage

As shown in the Figure: 3, a major group of students were confused and clueless about what to do next when they are using an advanced IDE for the first time. More than 75% of students needed continuous help even to get a simple programming task done. They needed guidance to understand and fix error messages, comprehend the functionality of code snippets etc. More than 85% of students have used introductory visual programming tools even once. Moreover, around 50% students stated that they are afraid to shift from using introductory programming tools to advanced programming tools.

A series of seminars was conducted to introduce CodeMage to school and undergraduate students, precisely to 83 students. We observed their adaptation to CodeMage and their reactions for the functionalities in CodeMage. Then we conducted a survey to get the effect of CodeMage. The results as shown in Table: 1 showed that more than 85% of novice programmers were more confident about their future in programming. A majority of students stated that CodeMage helped them to eliminate the fear of error messages and find clues about what to do next.

Even though there were a few who did not say yes to some questions it was due to the reason that they thought CodeMage will make a programmer's life too easy. This was then later justified by reminding them that the system is meant to motivate novices mainly.

Question	Yes	No
Did real-time feedback on errors help you fix errors fast?	76	7
Did you write your codes faster using code generator?	82	1
Did you find the friendly error messages helpful?	83	0
Was guidance to fix errors accurately and could you fix your errors easily?	81	2
Did the visual debugger help you to understand how program ran?	83	0
Did the visual debugger help you in reading the code?	80	3
Did you feel the need to have a remote tutor?	75	8
Did you like to have the CodeMage character in your programming environment?	79	4
Did you feel helpless while coding?	82	1
Do you recommend this tool to your friends?	83	0

Table 1: Results of survey done on CodeMage users

V. Conclusion

CodeMage is a new concept which provides a platform to the novice programmers for practicing Java programming. This will be a novel learning experience to the novice programmers all over the world. Here our main target group is novices who are with little knowledge or experience in programming. The main focus of this product extends to providing an educational programming environment which encourages and implant best programming practices and to eliminate the conflicts novices face when adapting to advanced tools and concepts. Therefore, the composition of tools and components within the development environment is carefully done.

CodeMage has its inbuilt editor which interacts with the user through an assistive tool which narrates syntax and logical errors in real-time. Cryptic error messages are converted into more user understandable language with more details when displayed to the user which gives the capability to trace back through the error message and find the exact location of the error. It encourages novice programmers to implement best programming practices by presenting hints in order to improve their coding styles while improving the efficiency and reduce

time consumption of novice programmers by introducing the code generation tool. The user-friendly database manipulation tool makes the CodeMage more unique. The visual debugger inbuilt with CodeMage visualize the step by step execution of the selected code snippet.

Feedback from users continuously indicated that this approach is reducing the gap for novice programmers to adapt from novice programming tool to advanced and complex programming. As future work, we intend to add more contextual information on errors to assist users and more hints to assist with correction of errors. Further studies are necessary to evaluate the usability and quality of the system with larger groups of users. CodeMage has more to give to the world one aspect which the team will look at is the security aspect of the product. The product was developed focusing mainly on simplicity speed and accuracy. The next big concern would be to enhance the security of the IDE as java can access many system functions these request should be controlled. The database manipulation tool can be extended to allow users to join tables with the use of visual simulation. The long-term goal would be the implementation of multiple programming language support.

VI. REFERENCES

[1] M. Prensky, "Programming is the new literacy", *Edutopia.org*, 2008. [Online]. Available: http://www.edutopia.org/literacy-computer-programming. [Accessed: 02- Mar- 2016].

[2] A. Robins, J. Rountree and N. Rountree, "Learning and Teaching Programming: A Review and Discussion", *Computer Science Education*, vol. 13, no. 2, pp. 137-172, 2003.

[3] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin and J. Paterson, "A survey of literature on the teaching of introductory programming", *SIGCSE Bull.*, vol. 39, no. 4, p. 204, 2007.

[4] P. Miller, J. Pane, G. Meter and S. Vorthmann, "Evolution of Novice Programming Environments: The Structure Editors of Carnegie Mellon University", *Interactive Learning Environments*, vol. 4, no. 2, pp. 140-158, 1994.

[5] M. Butler and M. Morgan, "Learning challenges faced by novice programming students studying high level and low feedback concepts", *Proceedings ascilite Singapore*, no. 99-107, 2007.

[6] D. Hagan, "Using BlueJ to teach Java (poster session)", *SIGCSE Bull.*, vol. 32, no. 3, pp. 188-189, 2000.

[7] E. Allen, R. Cartwright and B. Stoler, "DrJava", *SIGCSE Bull.*, vol. 34, no. 1, p. 137, 2002.

[8] J. Cross and T. Hendrix, "jGRASP", *SIGCSE Bull.*, vol. 38, no. 3, p. 356, 2006.

[9] "Blockly | Google Developers", *Google Developers*, 2016. [Online]. Available: https://developers.google.com/blockly/. [Accessed: 05- Jul- 2016].

[10] M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum and J. Silver, "Scratch", *Communications of the ACM*, vol. 52, no. 11, p. 60, 2009.

[11] *Alice.org*, 2016. [Online]. Available: http://alice.org. [Accessed: 01- Mar- 2016].

[12] Resnick, M. (2007). Learning from Scratch, Microsoft Faculty Connection, June 2007.

[13] M. McGrath and J. Brown, "Visual Learning for Science and Engineering", *IEEE Comput. Grap. Appl.*, vol. 25, no. 5, pp. 56-63, 2005.

[14] Jussi Kasurinen, Mika Purmonen and Uolevi Nikula "A Study of Visualization in Introductory Programming" Laboratory of Information Processing, Lappeenranta University of Technology, 2008

[15] D Weintrop, U Wilensky, "To Block or not to Block, That is the Question: Students' Perceptions of Blocks-based Programming," IDC '15 Proceedings of the 14th International Conference on Interaction Design and Children '2015, pp 199-208

[16] "GameMaker | YoYo Games", *Yoyo Games*, 2016. [Online]. Available: http://www.yoyogames.com/gamemaker. [Accessed: 06- Jun- 2016].

[17] Colleen M. Lewis, "How Programming Environment Shapes Perception, Learning, and Goals: Logo vs. Scratch," Proceedings of the 41st ACM technical symposium on Computer science'04, 2010, pp. 65-68.

[18] Michael Kolling, "The Greenfoot Programming Environment," ACM Transactions on Computing Education, v10, no. 4, Nov., pp. EJ908803, 2010.

[19] Scratch.mit.edu, "Scratch - Imagine, Program, Share", 2016. [Online]. Available: https://scratch.mit.edu. [Accessed: 01- Mar- 2016].

[20] Pane, J. Myers, B.A., & Miller, L.B. (2002). "Using HCI Techniques to Design a More Usable Programming System,"IEEE VL/HCC, 198-206.

[21] Murphy, L., Fitzgerald, S., Hanks, B., & McCauley, R. (2010)," Pair debugging: a Trans active discourse analysis,"ICER, 51-58.

[22] "Personifying programming tool feedback improves novice programmers' learning", in Proceedings of the seventh international workshop on Computing education research, 2011.

[23] Code.org, "Every child deserves opportunity", 2016. [Online]. Available: http://code.org. [Accessed: 01- Mar- 2016].