Computer Science and Engineering 2013, 3(3): 67-75
DOI: 10.5923/j.computer.20130303.03

Novel Assessment of Different Intelligent Tools for Problem Solving

Seyyed Meisam Taheri[1,*], Yamamoto Hidehiko[2], Hrudaya Kumar Tripathy[3]

[1]Mechanical and Civil Engineering Division, Graduate School, Gifu University, Yanagido, Gifu-shi, 501-1193, Japan
[2]Department of Mechanical Engineering, Faculty of Engineering, Gifu University, Yanagido, Gifu-shi, 501-1193, Japan
[3]School of Computing & Technology, Asia Pacific University of Technology & Innovation, Technology Park Malaysia, Kuala Lumpur, Malaysia

Abstract Teaching programming to novices is a difficult task down to the complex essence of the subject, the negative views associated with programming, and because initial programming courses are not often successful in encouraging students to understand the concepts. Foundation programming lessons should concentrate on problem-solving skills and introduction to the basic manners of algorithmic thinking. This paper's aim illustrates the progress and results obtained by investigating the different existing programming solving tools in order to achieve a new tool with high-performance capability. By using the intelligent visual tool, a user can comfortably analyze the problem and enhance the problem solving skills. This literature study's aim is to present a brief overview of the programming difficulties faced by novice students and of existing visualization tools in programming education.

Keywords Program, Intelligent Tool, Algorithm, Programming, Analysis, Coding

1. Introduction

The main reason of programming weakness in some new programmers is hidden in programming solving skill and techniques. The lack of understanding the programming concept comes from difficulties and complexities related to the programming environment and language syntax which are needed to use for novices[29]. According to a survey done by Lahtinen et al. most difficulties in programming that students face are due to the lack of understanding of how to design a program, how to solve a certain task and how to find bugs in their own programs[36]. Programming languages are usually considered as complicated and regularly have the most dropout rates[1]. Programming is a complicate work and lecturer must manage their teaching manner cautiously [23]. As a result beginners face various problems when they learn to program[1][37][38]. Some of these difficulties are:

- Installing and setting class paths for compilers.
- Learning functionalities of programming editors.
- Understanding programming questions and using programming language syntax knowledge to write code.
- Describing the program logic and the difficulty of translating logic to program.
- Poor quality of assistance offered by teachers.
- Lack of useful information about library functions and header files.
- Comprehending compiler error messages.
- Fix the errors, as determined during the debugging process.

Although several tools have been discussed in the literature review, comparison will be employed to find out which one implies to better comprehension for novice programmers. Indeed, some tools exist which are related to problem solving, especially in programming fields. This study is generally distributed entirely the literatures and comparison exist tools such as Jeliot, B#, Codewitz, Web Based Programming Assistance Tool for Novices (WPAT), E-Learning For Novice Programmers (A Dynamic Visualisation and Problem Solving Tool) and Flowcharts Interpreter (FI).

1.1. AIM

This study investigates the efficiency of using techniques and tools to achieve problem solving tools and to find out if such technologies and tools can aid novices in overcoming the current difficulties in programming by using problem solving tools and techniques. This study aims to find the best solutions and answers for the given problems, and inquiry will be asked via tool by users; Furthermore, it will give a general idea to author for designing a tool for novices to allow them to create flowcharts, convert flowcharts to pseudo-code and target programming languages. It is also important that Graphic user interface (GUI) should encourage a user to employ programming to solve the given problems. The mentioned tool will be integrated with a social

* Corresponding author:
s3812005@edu.gifu-u.ac.jp (Seyyed Meisam Taheri)
Published online at http://journal.sapub.org/computer

network; it will also uncover some related answers by using software agents that are in accordance with the user's interests. The Google services can help to improve this tool in searching with appropriate results, translating the problems, answers and pages' text, as well as, ranking the answers – all being functions in this tool.

2. The Iconic Programming Tool (B#)

In fact, the Iconic Programming Tool (B#) is a tool for beginner Programmers. The significant aim of design B# refers to difficulties faced in preliminary programming courses by novices, including problem-solving techniques and strategies, misconception about constructed programming languages, concepts and the traditional programming environments. An approach towards aforesaid complexities in programming by student programmers in preliminary programming lessons is the form of teaching. There are some strategies used to overcome the difficulties, one of which uses visual languages combined with the iconic flowchart method. In addition, iconic programming usually tries to make things easier in programming tasks by decreasing the level of accuracy and manual typing in programming languages. Thus, B# has provided an environment which aids programmers in programming by using iconic flowcharts. Basic programming concepts such as assignments, conditions, (inputs & outputs) and loops are supported in B#. Automatic generating codes, debugging and program executing are supported by the system as well. This study investigates B# structure, focusing on the aims that were B# followed as an iconic programming tool[14].

According to Hilburn points, students start programming with simple examples and statements, which do not aid students in increasing problem-solving skills which are vital for effective programming. He explains this method as a bottom-up approach and students should know the logical processes at first (top-down approach)[15]. The following table shows steps referred as in the program development Lifecycle, the tasks of students or novices who are trying to overcome the difficulty of processes of programming. This table shows that the programmer should implement the an algorithm; a visual environment instead of a text-based environment in this case can help the programmer to understand what to do and overcome the errors and syntaxes. There is too much focus on syntax, not sufficient emphasis on problem- solving and absence of support for experiencing program execution[9].

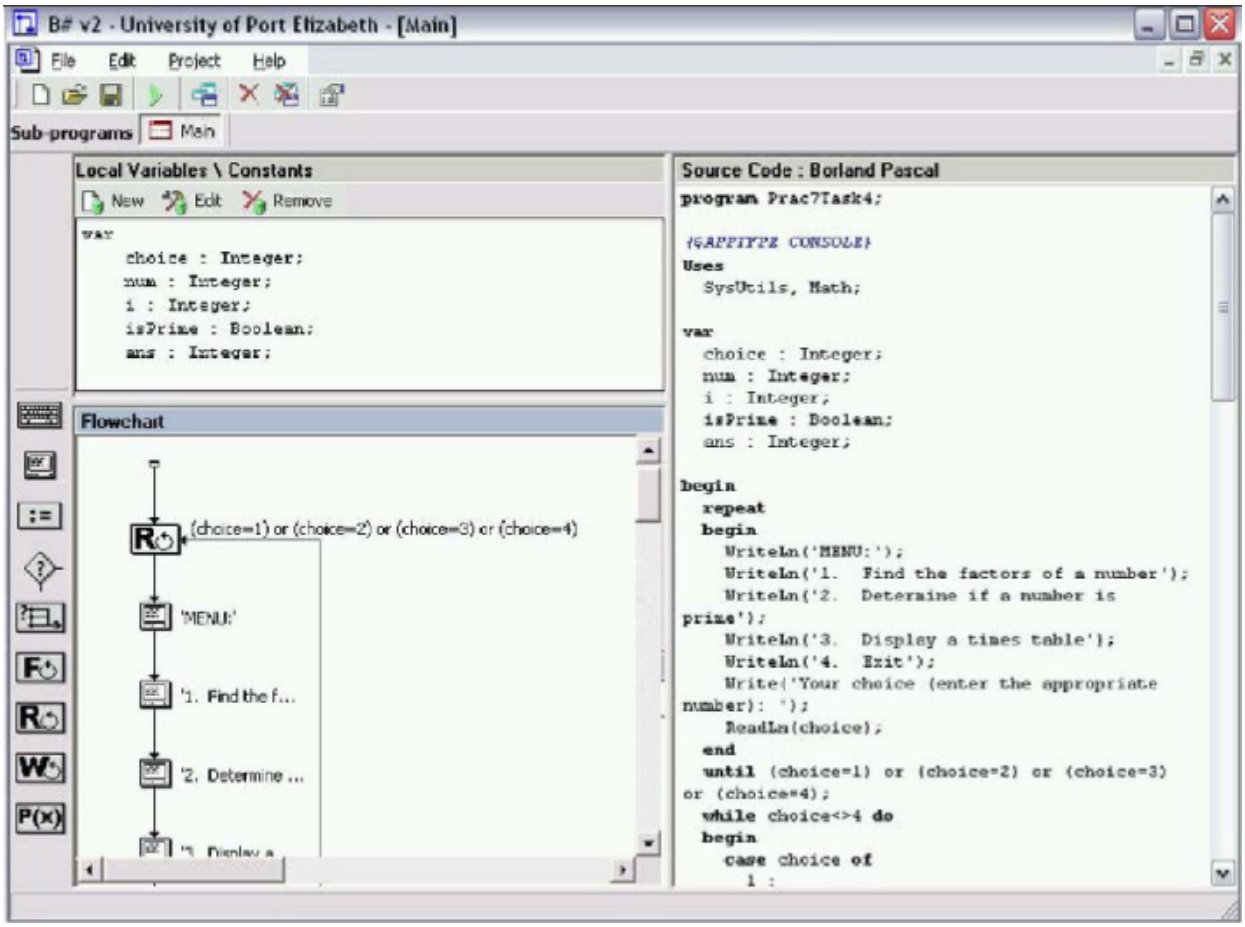

Figure 1. B# Environment[14]

The developers of B# have researched to help students to develop better understanding programming concepts by using iconic programming tools. The research is performed concerning the efficiency of B# as a teaching tool in an initial program[8]. Programmers need to have instant intelligent feedback about the correctness of the written program. Some evaluation sections can aid programmers to define the semantic correctness of the programs; the same module could provide teachers with the ability of identify defined features, which are needed to correct the students' programs.

Table 1. Development life cycle[14]

Step	procedure
1	Analyze the problem
2	Design a solution plan
3	Construct an algorithm
4	Implement the algorithm
5	Test and debug the algorithm

2.1. B# Justification

The research shows that novices prefer a visual and iconic environment. For instance, B# Programming environment is a visual environment instead of a text-based environment. In this case, that can help the programmer to understand what to do as well as errors and syntaxes. The tool which is our aim to achieve in some case is following B# structure. The tool uses an iconic and visual environment by using the best metaphor and tips, for better understanding of each part, and its errors and difficulties. The B# tool might have been the perfect idea and tool in its period, but taking a first look at it, the user will tire of working with this tool. Nowadays, the most significant need which each tool in this era of technology might have is to be interactive and exciting. In fact, the idea and the design of B# is very simple, but the outcome could be that much more effective. The same idea which is used in B# will be used in our tool as well.

3. E-Learning (Visualisation and Dynamic Tool)

Programinate is an environment based on the web that aims to conquer the issues faced by novice programmers. The concentration of this tool is using flowcharts to find solutions to fundamental programming problems and provide visualization for programming to correct generated code of the given program. New programmers also have difficulty to understand the problem specification and converting the problems to code. The difficulties are related to programming environment, syntaxes and non tracing abilities that make the programming more difficult for novices. The results show attention to algorithmic problem-solving and development is fewer[29].

The best form of visualization for programmers, especially novices is Flowchart. It has been a long time since flowcharts have been used to visualize the structure of programs. The flowcharts are quite easy to understand without having any background in programming - with using algorithm beside flowcharts, the level of understanding will increase. Westphal says that "without the use of diagrams or flowcharts, it is difficult for beginners, even with pseudo code to communicate the flow of a program"[33]. Ben-Bassat says assuming that dynamic animation can expand the flowchart's effectiveness as a novice assist in algorithmic problem solving and program development[5].

3.1. The Programinate Environment

The tool aims to provide an environment with visual form and the program that runs for encouraging the novices, aims to be a tool as long as they need an assistant in programming. The code generation part is provided with two different languages presently; Java and two types of visual basic. The tool uses colors to separate different parts such as types, components, flowcharts, codes and etc. The program uses trace panel, which shows the variable values, and variable changes, which help to understand the process. The results of feedback which have gained by observation, interview and questionnaire:

Table 2. Likert Questionnaire Responces[29]

Cat	Aspect	15-17	13-15	Uni	Total	
		Score	*Score*	*Score*	*Rsp*	*Score*
U1	It was easy to use.	60.71	71.21	78.6	131	74.75
U2	It looked easy to use.	50	59.85	72	131	66.49
U3	It was enjoyable.	55.36	75	66.6	128	67.53
U4	It looked interesting.		75	67.5	116	69.61
U5	It was reliable.	66.67	69.12	75	72	70.60
U6	It was speedy and responsive.	67.86	75.9		47	73.51
U8	It is suitable for beginners.	59.62	81.45		45	74.66
E1	I learnt something using it.	62.5	75.78		46	71.74
E2	It was useful in my studies.			72.5	81	72.53
E3	It enhanced my understanding of programming			69.6	23	69.57
E4	The flowcharts aided my comprehension.	62.5	71.69	74.3	72	70.78
E5	The use of color in the flowcharts was beneficial.	71.15	71.32	77.2	71	73.18
E6	The relationship between flowchart and code was clear.	48.21	74.05	75	71	69.26
E7	I gained some understanding of the code generated.	51.79	71.21		47	65.43
E8	The animation features were helpful.		68.38	77.5	54	71.76
P1	The problem solving exercises were fun.	53.57	75		47	68.62
P2	The problems were at the right level of difficulty for me.	66	69.85		48	68.73

- Usability
  U questions review ease of use, and how much the tool is usable and enjoyable for programmers.
- Efficacy
  The E questions are about tool efficacy as a teaching and helping tool. The results illustrated that Programinate is generally helpful in age between 13-15. Additionally, the results show that inspection features and animations were mainly helpful.
- problem solving exercises
  The section shows the enjoyment of using the tools during the solve problems. The results illustrated that 13-15 age groups have seen the tool as enjoyable.

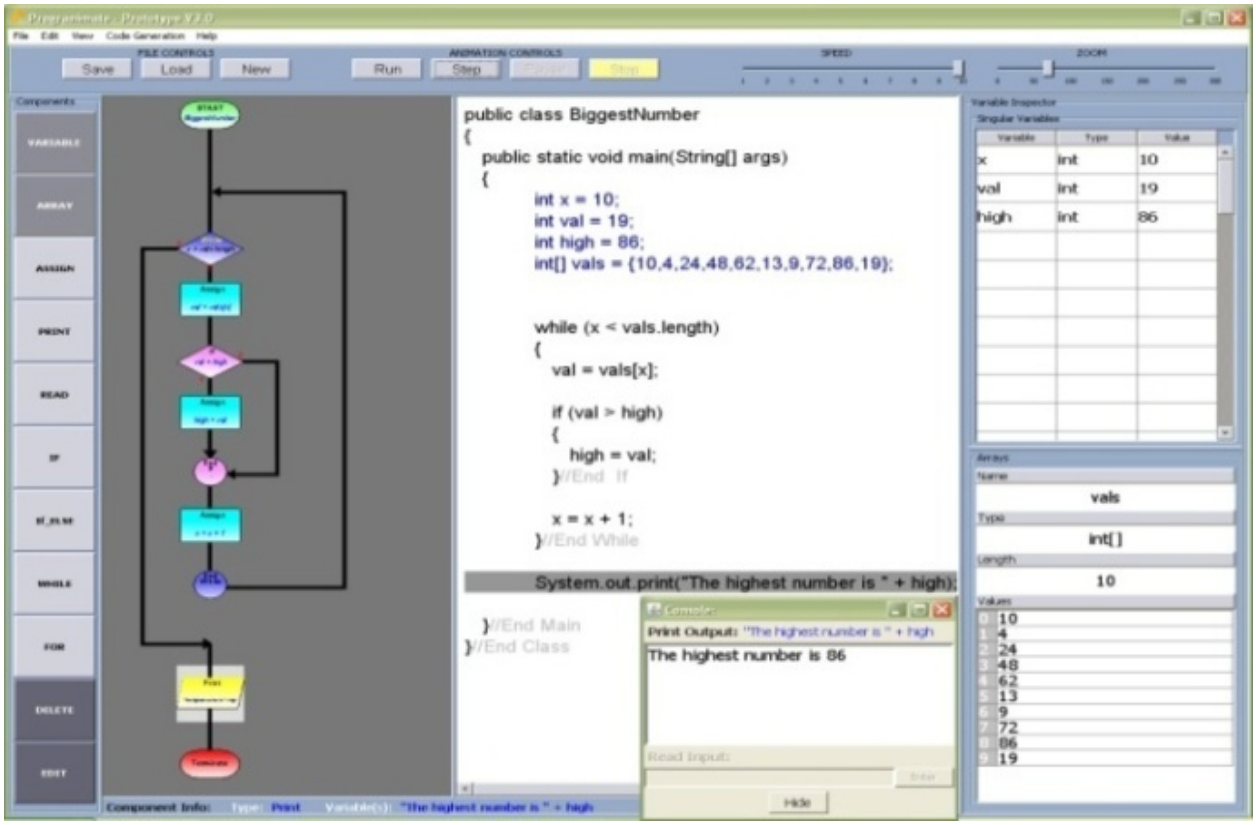

Figure 2. The Programinate Environment[29]

3.2. E-Learning Justification

The visualization of programming milestones is the environment of the tool which is provided by the flowcharts, coding, tracing or allocating the variables. The Programinate tool is efficient and usable according to the questionnaire which is provided in the last researches pertaining to this tool for accomplishing, analysis and testing the tool. In some cases, the environment is very user friendly; however it is very simple in the first glance. This kind of tool can be very helpful to students, because the novice students can understand what is going on in the program without leaving the current page, and all the environment features are located in one place, which makes the tool more understandable. In fact, novices' difficulty in programming related to points that they do not understand-especially where and how the variable initialized. If they able to see the process of the programming and tracing the code, that can be much easier to understand the concept[29].

4. Software Visualization Tool: Jeliot

The Jeliot tool is intended to assist novice programmers in learning object oriented and procedural programming. The best part of Jeliot is semi-automatic visualization and control flows. Generally, when students want to learn programming, this type of tools can be a brilliant learning source. The idea behind Jeliot is to encourage students to construct their own programs and give them permission the ability to see the visual illustration of the program execution. These processes help them to develop their mental model about calculation that assists them in understand the concept of programming. Therefore, the programmers are engaged with the tool and since they are working with the tool they are also learning. The object-oriented models in visual mode are very significant - for that reason these concepts are not easily comprehend by novices in programming[3].

The first developed tool to teach introduction object-orie nted programming is BlueJ[10]. The highlight in the system is the static visualization of the class structure, same as a UML diagram. Javavis is a system developed from the similar thought of using the Java Debugging Interface (JDI) to gain information about the runtime performance of the program[25]. In fact, this kind of tools could be very helpful for experts in programming as well.

4.1. Jeliot Goals

The aims of Jeliot 3 were described in studies of the earlier versions of the jeliot. The major aims of the systems are following below:

- Usability.
- The visualizations should be dependable in all cases.
- The visualizations should be comprehensive and incessant.
- The system should support the visualization of as large a subset of programs written in the Java language as possible.
- The system should be extendable internally and externally.

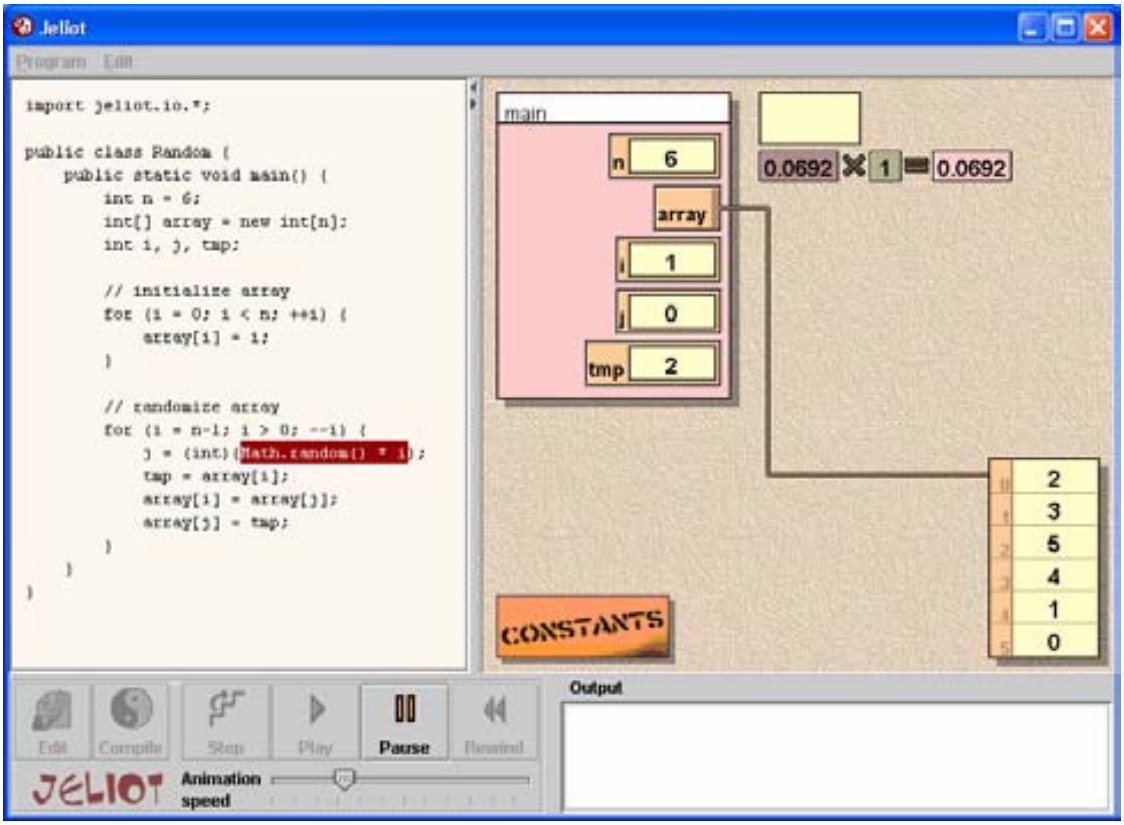

Figure 3. User interface of Jeliot 2000[3]

The first three goals come from the fact that Jeliot 3 is intended for novice users, and in research on Jeliot[24][19] and visual displays[20] these features have been found significant for them.

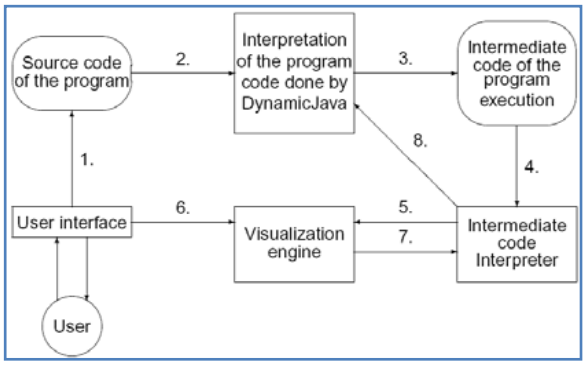

Figure 4. The structure of Jeliot 3[3]

Jeliot 3 can assist novices in first stages of programming by providing understandable semantics and by encouraging novices through the learning process. Teachers and students can use graphical and oral vocabulary that simplify the conversation of programming concepts.

4.2. Jeliot Justification

The jeliot environment is very simple and easy to use. The jeliot developers have tried to make it as simple as they can. The highlight of this tool is the visual environment and tracing which make it more user-friendly and understandable. The tracing part shows the process step by step, and this is a useful part of this tool. The tool also uses the java programming language as a default; also the code is compiled inside the tool and it will show any error in the program before run the program - it might be helpful in other aspects as well. The relation between our tool and jeliot can be the visualization, tracing and using the illustration all processors in one screen. Jeliot and other tools share something in common in that which they are trying to be more visual and easy to use; this is because the programming in simple and text-base tools can be boring.

5. The Codewitz Project

Robins et al. recommend that lecturers should concentrate on the combination and use of these features, especially on issues of basic program design[26]. For instance, structure visualization shows examples of the fundamental constructions and their combinations in different situations could encourage students towards better understanding of different approaches and construct a mental library of various answer diagrams for program design. The object oriented approach, pointers and memory are most difficult contents in visualizations, where Codewitz could be especially useful in overcoming these difficulties. The research on algorithm model and visualization in educational locations can offer important information for developing Codewitz simulations[21].

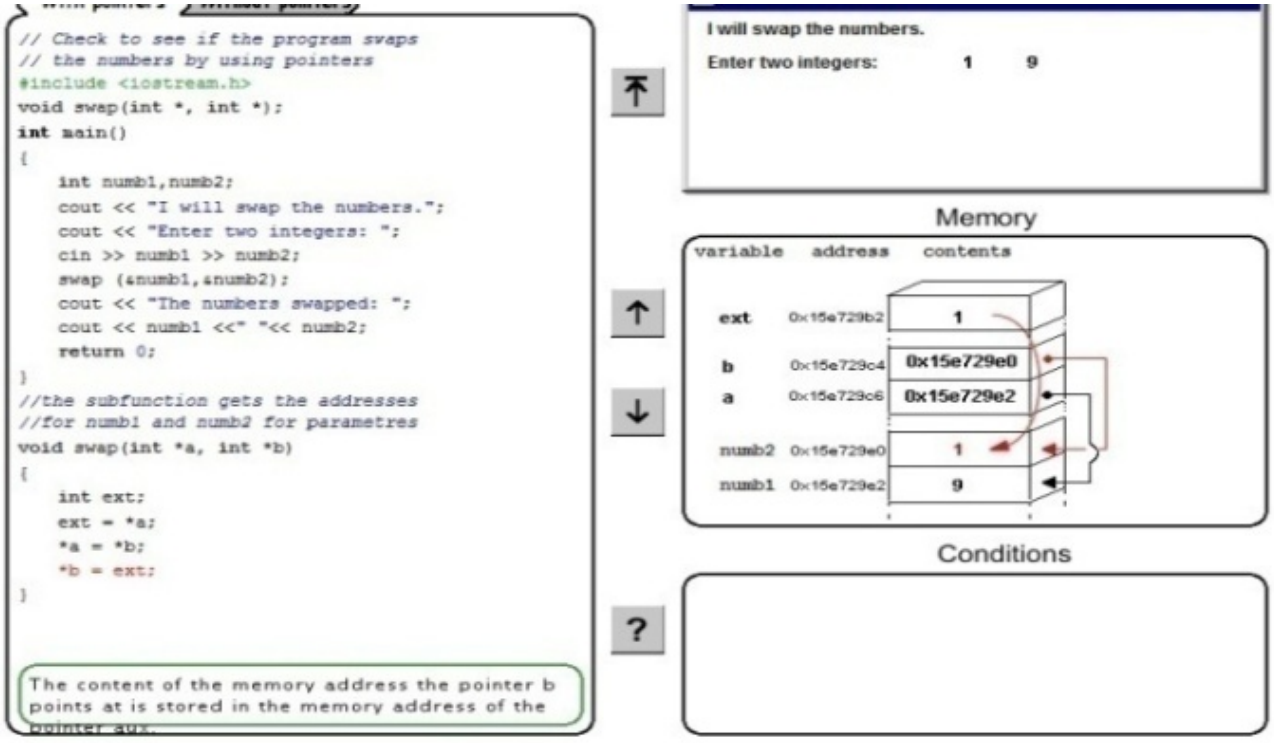

Figure 5. Codewitz environment

The main goals of the Codewitz-project are:

1. Developing and producing interactive learning objects for fundamental programming courses.
2. Providing a growing repository for string and sharing the resource with the project partners.
3. Creating a network of teaching experts who deal with this problem every day."

The Codewitz objects are individually mutual exercises, or assignments, for the students to utilize during study. Teachers also can use the tool to help them to teach. The tool gives students the ability to use at home or at school, and teachers as well. The tool's objects can have up to five areas or windows, Input/output, program execution, memory, condition and explanations parts as Figure 5 shows.

5.1. Codewitz Justification

The codewitz tool is used for students and teachers both for a better understanding about the programs and what their processes and variable memory allocations are. The environment is divided into different sections which make the tool more understandable; each part is responsible for specific operations. This tool in some case is same as jeliot 3 but with this difference that the variables allocation in the memory is in visual mode. This tool is very helpful which owe to the differences between commands, modules and operations and clarify when the tool shows step by step what is in the memory and how the variables, addresses and memory content are changed. The tool's manner and the result of using this tool are really impressive. Using the same techniques in a new tool will help novice programmers to have a deeper understanding of programming.

6. Raptor: Flowchart based Programming Environment

Research shows that "most students are visual learners and instructors tend to present information verbally; Studies estimate that between 75% and 83% of students are visual learners"[30][12]. RAPTOR the programming flowchart-based environment is designed purposely to assist students in facing syntactic bugs and visualize the algorithms for them. The RAPTOR program aids the student in executing programs visually and tracing the execution with following flowcharts. Most students prefer to use flowchart to declare their algorithms, and results show that RAPTOR helps them more than traditional language or writing flowcharts without RAPTOR. The Raptor environment also allows users to design an algorithm by using flowchart signs and combine them together. Users can run the algorithm and see the process step by step or in continuous mode.

Some reasons to use RAPTOR:

- The RAPTOR reduced the quantity of syntax that must be learned to write proper program.
- The RAPTOR is visual. Programs are diagrams that can be executed one symbol at a time. This helps to follow the flow of step execution into RAPTOR programs.
- RAPTOR is designed for ease of use.
- RAPTOR error messages are designed to be more readable and understandable for novice programmers.

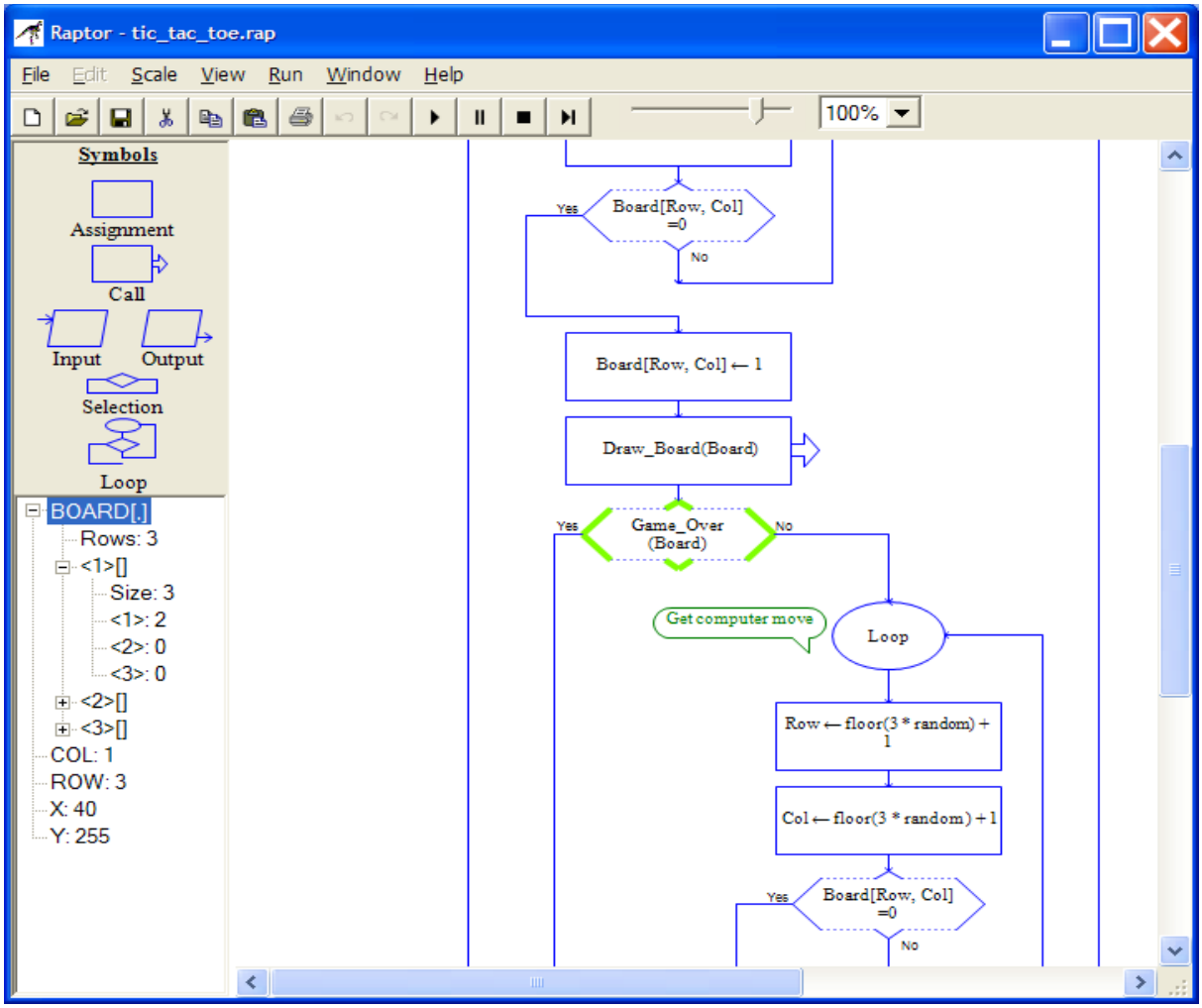

Figure 6. A RAPTOR flowchart in action[30]

6.1. Raptor Justification

Raptor is a kind of problem solving tools which is concentrate on flowcharting. To learn basic programming, learning flowchart and Algorithm is a must. Using the basic problem solving techniques in a new tool by following the architecture of RAPTOR will make the tool much more effective. We believe that the all kinds of users might have dealt with tool, novices, experts and those just familiar with laymen terms. Furthermore, the tool must consider different users with different ability and needs, for instance some users prefer to learn programming with flowcharts while others might think he or she does not need to focus on flowcharts or algorithms.

7. Software Agent

Software agents are independent parts of software that are responsible for several tasks dedicated to them. In the era of technology, the benefit is authorizing given types of tasks to run automatically by autonomous software programs. Software agents are continuously running, modified and semi-autonomous; also it constructs them in order to be helpful with a large variety of information and procedure management tasks. Software agent is software that functions endlessly and separately in an exacting environment, which may include one more agents and processes[2].

Intelligent agents continuously perform three functions as:[2]

- Observation of dynamic circumstances in the situations
- Act to influence circumstances in the environment
- Analysis to understand comprehensions, solves problems, draw deductive and resolve actions

7.1. Software Agent: Decision Making

The most important part of agents is to have the capability to make a decision. The agent also must make a decision on how to react as well as the most suitable time to answer instantly, or if time is needed to examine the situation. Furthermore, an active agent is able to take action without being purposely requested if it senses an appropriate situation. Obviously, this ability needs an agent to be capable to make a decision when to do an action as well as what action to do. In addition, apart from simply making a decision, all decisions are not excellent decisions. Thus protocols of decision making are regularly analyzed and

evaluated by factors such as: time, ease, constancy, social interests, Pareto efficiency, individual reasonableness, computational efficiency, distribution and communication efficiency. At negotiation time, it is obviously not helpful for an agent to take very long periods of time to make a decision; as such the decision making device cannot be used in sensible situations[2].

7.2. Agent Properties

A software agent is a computer system situated in an environment that performs on behalf of its user and is characterised by a number of properties[7]. Most researchers agree that autonomy is a crucial property of an agent. Furthermore, cooperation among different software agents may be very useful in achieving the objectives an agent has[7]. According to the weak notion of agency given by Wooldridge the most general way in which the term agent is used to denote hardware or (more usually) software-based computer system that enjoys the following properties: autonomy, social ability, reactivity and pro-activeness[34]. Identify three key concepts in their definition that they adapt from): situated, autonomy, and flexibility (by the term flexible they mean that the system is responsive, pro-active and social). An agent is a system that enjoys autonomy, social ability, reactivity and pro-activeness. He also said the fact that other researchers discuss that different properties, such as mobility, actuality, kindness, wisdom and learning, should receive greater importance.

7.3. Agents Justification

As mentioned above, agents are playing an important role in the new era of technology. In this particular tool which is employing to solve problems, and for each problem might have different solutions; therefore, the tool needs to be intelligent to find out which one is more suitable. In fact, the tool should be able to complete its knowledge base due to the information which experts and the high ranking users are imported to the system or update the previous data. The agent's task is solving problems, also the tool needs software that to make decisions in some situations.

8. Future Work and Conclusions

This research illustrates, how the problem solving tool can be useful and effective when used in an initial programming course by using Artificial Intelligent (AI), Software Agents, Search Algorithms, Google's services and Social networks but the potential will remain to investigate more methods for future works. We want to develop our tool to be more user-friendly, effective and useful by using the tool as a social network where the guidance in this tool would be the most expert users. The users' skills and position will be chosen by their ranking, and the first registration form which they need to fill in during the registration, and their qualification will be approved by the system administration. As well as, the ranking system will follow the Google ranking system. The tool can be used as a virtual classroom, the experts and lecturers can be the administrator of the class. In addition, the tool's focus point is on the initial phases of programming, and for future phases of programming novices do not allow to access. Also access to high levels needs the lecturers or agents' permission. Our review shows that visualization interfaces are more effective than the text-base learning system. In conclusion, we are trying to encourage novices and attract them with the visual tool to learn and do programming.

ACKNOWLEDGMENTS

This research would not be possible without the help of my professors. Thanks to all other members & friends directly or indirectly supported in this research and also for the future.

REFERENCES

[1] A. Robins, J. Rountree, and N. Rountree. Learning and teaching programming: A review and discussion. Computer Science Education 13(2): 137--172, 2003.

[2] Akshatha.P.S, Pooja Rani (2011). SOFTWARE AGENT'S DECISION MAKING APPROACH BASED ON GAME THEORY. International Journal of Advances in Engineering. 1, pp.10.

[3] Andrés Moreno, Niko Myller. Producing an Educationally Effective and Usable Tool for Learning, the Case of the Jeliot Family. To appear in the Proceedings of International Conference on Networked e-learning for European Universities, Granada, Spain, 2003.

[4] Ask.com Takes the Lead on Log Retention; Microsoft and Yahoo! Follow https://www.eff.org/deeplinks/2007/07/ask-com-takes-lead-log-retention-microsoft-and-yahoo-follow), eff.org, Retrieved on 2008-01-03.

[5] Ben-Bassat Levy R, Ben Ari M and Uronen P, "An Extended Experiment with Jeliot 2000", In Proceedings of the First International Program Visualization Workshop, University of Joensuu Press, Porvoo Finland, 2001, pp: 131-140.

[6] Cardellini, L. An Interview with Richard M. Felder. Journal of Science Education 3(2), (2002), 62-65.

[7] Chira, C. (2003). Software Agents, IDIMS Report, 2/21/03.

[8] Cilliers, C. "The Implementation of Alternative Delivery Modes in a South African Introductory Programming Course". PhD Thesis, Department of Computer Science and Information Systems, NMMU, Port Elizabeth, 2005.

[9] Crews, T and Ziegler, U. "The flowchart interpreter for introductory programming courses", Proceeding of the Frontiers in Education 1998 Conference. Tempe, Arizona, USA, 1996.

[10] D. J. Barnes and M. Kölling. Objects First with Java – A Practical Introduction using BlueJ. Prentice Hall/Pearson

Education, Reading, Massachusetts, USA, 2003.

[11] Does AskEraser Really Erase?" Electronic Privacy Information Center.. Retrieved 2008-03-10. (http://epic.org/privacy/ask/default.html).

[12] Fowler, L., Allen, M., Armarego, J., and Mackenzie, J. Learning styles and CASE tools in Software Engineering. In A. Herrmann and M.M. Kulski (eds), Flexible Futures in Tertiary Teaching. Proceedings of the 9th Annual Teaching Learning Forum, February 2000. http://ccea.curtin.edu.au/tlf/tlf2000/fowler.html.

[13] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references).*

[14] Greyling, J.H.; Cilliers, C.B.; Calitz, A.P.; , "B#: The Development and Assessment of an Iconic Programming Tool for Novice Programmers," Information Technology Based Higher Education and Training, 2006. ITHET '06. 7th International Conference on, vol., no., pp.367-375, 10-13 July 2006.

[15] Hilburn, T.B. (1993). A top-down approach to teaching an introductory computer science course. 24th SIGSCE Technical Symposium of Computer Science Education. Indianapolis, USA, 1993.

[16] Kolbrún Fanngeirsdóttir (2003). Konur og tölvunarfræði. On the internet 23.04.04 at http://www.vhr.is/kennarar/asrun/Efni/Skyrslakonur.pdf.

[17] Letter to U.S. Federal Trade Commission" (https://www.cdt.org/privacy/20080123_FTC_Ask.pdf) (PDF). Center for Democracy and Technology. January 23, 2008. . Retrieved 2008-03-10.

[18] M. Ben-Ari, N. Myller, E. Sutinen, and J. Tarhio. Perspectives on Program Animation with Jeliot. In S. Diehl, editor, Software Visualization, vol. 2269 of Lecture Notes in Computer Science, pages 31–45. Springer-Verlag, 2002.

[19] M. Lattu, V. Meisalo, and J. Tarhio. A visualization tool as a demonstration aid. Computers & Education, 41(2):133–148, 2003.

[20] M. Petre. Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. Communication of the ACM, 38(6):55–70, 1995.

[21] Milne, I., Rowe, G. (2002). Difficulties in Learning and teaching Programming - Views of Students and Tutors, Education and Information Technologies, 7(1), pp. 55-66.

[22] Myller, N.; Bednarik, R.; Moreno, A.; , "Integrating Dynamic Program Visualization into BlueJ: the Jeliot 3 Extension," Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on, vol., no., pp.505-506, 18-20 July 2007.

[23] Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C.,Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Velázquez-Iturbide, J.Á. (2003), Exploring the role of visualization and engagement in computer science education, SIGCSE Bulletin, 35(2),pp. 131-152.

[24] R. Ben-Bassat Levy, M. Ben-Ari, and P. A. Uronen. The Jeliot 2000 program animation system. Computers & Education, 40(1):15–21, 2003.

[25] R. Oechsle and T. Schmitt. JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI). In S. Diehl, editor, Software Visualization, volume 2269 of Lecture Notes in Computer Science, pages 176–190. Springer-Verlag, 2002.

[26] Rist, R. (1996). Teaching Eiffel as a first language. Journal of Object-Oriented Programming, 9, pp. 30-41.

[27] Shackelford, R., and LeBlanc, R. Introducing Computer Science Fundamentals Before Programming. Proceedings of FIE '97, 285-289.

[28] Scott A, Eyres D and Watkins M, "A Step Back From Coding- An Online Environment and Pedagogy for Novice Programmers", Proceedings of the 11h Java in the Internet Curriculum Conference, The Higher Education Academy, London Metropolitan University - UK, 2007, pp: 35-41.

[29] Scott, A.; Watkins, M.; McPhee, D.; , "E-Learning For Novice Programmers; A Dynamic Visualisation and Problem Solving Tool," Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on, vol., no., pp.1-6, 7-11 April 2008.

[30] Thomas, L., Ratcliffe, M., Woodbury, J. and Jarman, E. Learning Styles and Performance in the Introductory Programming Sequence. Proceedings of the 33rd SIGCSE Symposium (March 2002), 33-42.

[31] Ulle Endriss. Multiagent systems: Rational decision making and negotiation. http://www.doc.ic.ac.uk/ue/mas, 2005.

[32] United States Patent Database (http://patft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=7,047,502.PN.&OS=PN/7,047,502&RS=PN/7,047,502), US Patents, 2006-06-16.Retrieved on May 16, 2006.

[33] Westphal B, Harris F and Fadali M, "Graphical Programming: A Vehicle for Teaching Computer Problem Solving", 33rd ASEE/IEEE Frontiers in Education Conference, IEEE, Boulder Colorado, 2003, pp: 19-23.

[34] Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: theory and practice. The Knowledge Engineering Review 10(2), 115-152.

[35] Zogheib Ali: Automatic Language Translation - Statistic-based System. REPORTNO. 2007:4. Chalmers University of Technology, Sweden.

[36] E. Lahtinen, K. Ala-Mutka, and H. M Jrvinen, "A study of the difficulties of novice programmers", in Proc. 10th annual SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSEOS), New York,USA, pp.14-18.

[37] N. Truong, "A web-based programming environment for Novice Programmers," Ph.D. dissertation, Faculty of Inform. Technology, Queensland University of Technology, Queensland, 2007.

[38] V. G. Renumol, S. Jayaprakash and D. Janakiram, "Classification of cognitive difficulties of students to learn computer programming," Indian Instit. of Technology, Depart. of Comput. Sci., Chennai, 2009.