PROGRAMMING ENVIRONMENT FOR TEACHING INTRODUCTORY PROGRAMMING FOR SECONDARY SCHOOL

NUR MARDHATI NABILA BINTI TUKIMAT

A dissertation submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Computer Science

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

MARCH 2019

DEDICATION

This thesis is dedicated to my father, who taught me that the best kind of knowledge to have is that which is learned for its own sake. It is also dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time.

ACKNOWLEDGEMENT

In the Name of Allah, The Most Gracious and The Most Merciful.

First and foremost, all praise to The Almighty, all praise to Allah for granting me strength to complete my master study after two years effort and tears. It my pleasure to acknowledge all my friends and colleagues for their enormous supports and sharing ideas in accomplish this study. I will always be grateful to have friends that are always being my sides. In preparing this thesis, I was in contact with many people, researchers, academicians, teachers and students. They have contributed towards my understanding and thoughts.

My sincere appreciation to my supervisor, Dr. Dayang Norhayati, for the informative supervision and countless hours spent in sharing and understanding. Her depth knowledge on a broad spectrum of computer science has extremely beneficial for my project and improvised my knowledge in this field. My deeply thanks to Ayah, Mama, Sayang, Kak Ayu, Kak Anne, Bazli and Dayah for encouragement, guidance, valuable comments and motivation. And also, for my little princess; Aisyah and Qisya, thanks for your understanding and love.

ABSTRACT

Programming subject was become one of the syllabus in Malaysia national school start from 2017. To introduce youth to programming, suitable programming environment to teach introductory programming should be determined. Many initiatives are proceeding to bring powerful ideas of computing into classroom around the world. A popular strategy being employed in this effort is the use of block-based programming environment. This environment found to be effective among younger learners. Their suitability in high school context is an open question. The existing tools was analysed to identify the suitable environment to teach introductory programming in high school. An experiment involving 30 participants was conducted to get their perception on three different programming environments; text-based, block-based and hybrid. Findings from the study reveal that participants in hybrid group scoring highest in content assessment and reporting higher level in enjoyment and engagement to traditional programming structure. After the completion of literature and exploratory research, a bidirectional hybrid programming environment was developed. This environment combines features of block-based and text-based interface to provides the platform and engagement of block-based tools with the power and authenticity of text-based introductory environment. A traditional hybrid programming creates a gap between block-based and text-based programming. It was be used to run in evaluation workshop involving of 13 students aged 16 – 17 years old. The evaluation of enhanced programming environment was determined by using triangulation of data; students' perception and result from their assessment using an enhanced hybrid programming environment. Participants have positive perception on confidence and understanding of programming concept. Besides, they agreed that bidirectional hybrid programming environment offered a more effective way of introductory programming subject compared to existing environment they are using in classroom. Suggestions for future work are outlined and intended that this research will assist the development and use a bidirectional hybrid environment in teaching introductory programming.

ABSTRAK

Pengaturcaraan merupakan salah satu subjek di dalam sukatan pelajaran sekolah-sekolah di Malaysia bermula tahun 2017. Untuk memperkenalkan belia kepada pengaturcaraan, persekitaran pengaturcaraan yang sesuai perlu digunakan. Terdapat tiga jenis persekitaran pengaturcaraan seperti berasaskan teks, berasaskan blok dan hibrid kombinasi teks dan blok. Para penyelidik mendapati bahawa penggunaan pendekatan dan persekitaran yang betul dapat menjadikan proses pembelajaran lebih efektif dan meningkatkan minat pelajar ke atas subjek pengaturcaraan. Terdapat beberapa masalah dalam penyediaan mengajar subjek ini termasuk kurikulum, pemilihan Bahasa, pedagogi pembelajaran, dan persekitaran pengaturcaraan untuk menyokong pembelajaran. Untuk mengenal pasti persekitaran yang sesuai untuk mengajar pengenalan kepada pengaturcaraan, perisian sedia ada dikaji dan dianalisis. Eksperimen melibatkan 30 peserta dijalankan untuk mendapatkan persepsi pelajar terhadap persekitaran pengaturcaraan berasaskan teks, berasaskan blok dan hybrid. Hasil kajian mendapati peserta bagi kumpulan hibrid mencatat rekod tertinggi dalam penilaian komutatif dan merasa sangat mudah untuk beralih ke pengaturcaraan berasaskan teks. Setelah selesai kajian kesusastreraan dan penyelidikan, persekitaraan pengaturcaraan hibrid dua arah telah dibangunkan. Ia digunakan dalam bengkel penilaian melibatkan 13 orang pelajar berusia 16 – 17 tahun. Keberkesanan ditentukan dengan kaedah triangulasi data; persepsi pelajar dan keputusan pengaturcaraan pelajar setelah menggunakan persekitaran hibrid yang telah dipertingkatkan. Peserta memberi perseptif positif terhadap keyakinan dan pemahaman konsep pengaturcaraan. Selain itu, mereka bersetuju bahawa persekitaran pengaturcaran hibrid dua hala menawarkan cara yang lebih berkesan bagi subjek pengaturcaraan berbanding persekitaran yang sedia ada yang mereka gunakan di kelas. Cadanngan untuk kerja akan datang digariskan dan bertujuan agar penyelidikan ini dapat membantu pembangunan dan penggunaan persekitaran hibrid dua hala dalam pengajaran pengaturcaraan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

MDEC	-	Malaysian Digital Economy Corporation
HTML	-	Hypertext Markup Language
IDE	-	Integrated Development Environment
ASK	-	Asas Sains Komputer
SK	-	Sains Komputer
SDLC	-	Software Development Life Cycle
JRE	-	Java Runtime Error
JDK	-	Java Development Kit
UTM	-	Universiti Teknologi Malaysia
SMK	-	Sekolah Menengah Kebangsaan
SaaS	-	Software as a Service
PaaS	-	Platform as a Service
IaaS	-	Infrastructure as a Service

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

Since the Stone Age, humans tend to make their lives easier by inventing variety of tools. In the twenty-first century, devices and gadget that solve our daily lives problems were created from mechanical, electronics and glued together with computer program. The first computer program was written by Ada Lovelace in 1842 using Analytical Engine invented by mathematician Charles Babbage for calculating Bernoulli numbers. In 1951, an American computer scientist Grace Hopper wrote the first compiler known as A-0 in UNIVAC to convert sequences of subroutines and arguments into computer. Hopper works lead the computer program to be more human readable and, in 1957 IBM invented the first major programming language called FORTRAN that introduce the usage of IF, DO and GOTO statements.

In this century, one of the useful products of technology would be the computers. It becomes indispensable in our daily lives and increase the demand for computer scientist, which is expected to increase irrespective of the poor current state of the economy (Wellman et al, 2009). There is a necessity to produce competent computer programmer not just to program PCs and PDAs, but also washing machine, microwaves and a range of another essential item. However, there is a problem to teach students that do not have ability to code a program and attract their interest of study of programming (Bergin, 2006).

In Malaysia, programming subject become one of the syllabuses in national school starting 2017. According to CEO of Malaysian Digital Economy Corporation (MDEC) Datuk Yasmin Mahmood, this subject incorporated into the teaching method, especially in science and mathematics classes. This is an effort of government to encourage youth to take part in technology making instead of just being a user.

Besides, this effort encourages cognitive and higher thinking skills among youth (Malay Mail Newspaper, July 18, 2016). In addition, primary and secondary school students were also exposed to computational thinking skills that be integrated into school curriculum (New Straits Times, August 11, 2016). According to Wing (2006), computational thinking is an approach for problem solving, system designing, and human behaviour understanding that extracts on the power and limits of computing. In addition, Wing (2006) claimed that this skill is an essential for everyone and should add to every child's analytical ability besides reading, writing, and arithmetic skill.

There are two distinct standards under Ministry of Education Malaysia which are Primary and Secondary. Primary level (Standard 1-6) is a student's age 7 to 12 years' old, while secondary level (Form 1-5) is a student's age 13 to 17 years' old. Primary level will be taught coding where school syllabus will be integrated with computational thinking. This lead to improve problem solving and critical thinking skills. At this level, student will be introduced to programming environment such Scratch so that students can applied their skills to practical situations. On the other hand, when they get into secondary level, they will be exposed to more advanced programming using programming language such Java and Hypertext Markup Language (HTML). Learning coding not only gives students knowledge to program a computer, but it builds problem solving skills, creative expression, and development of computational thinking.

There is significant debate about how to teach programming to novice since the introductory programming courses introduced in school and colleges. A lot of things need to be considered when constructing the course such curriculum, languages choices, learning pedagogy, and programming environment and tools for supporting learning (Pears et al, 2007). The term programming environment is referring to an environment that contains language specific editors and source level debugging facilities (Jones, 2004). This term also known as integrated development environment (IDE) that provides comprehensive facilities to computer programmers for software development. An environment might contain a text editor (for program preparation), an assembler (for program translation to machine language), and a simple operating system.

1.2 Research Background

Anyone who wants to learn programming must choose a programming environment to create and run the program. Programming environment can be divided into three different types which are text-based, block-based and hybrid blocks/text programming. Text-based environment is an environment which the primary input and output based on text rather than visual or sound. In addition, users require to comply and conform to the formal syntax of the programming language.

Two decades ago, MIT Media Lab introduced a concept of block-based programming that eliminated the need to learn and memorize the syntax of a formal programming language (Schor, 2016). A block-based programming environment consists of variety of visual that leverage a puzzle metaphor and support drag-and-drop approach. This mean, users only use a mouse to assemble functioning program by snap together the instructions. Finding from research done by Weintrop & Wilensky (2015), block-based programming environment ease-of-use especially among high school students. There are many factors contribute to this statement; block using natural language, interaction of drag-and-drop composition, and the ease of browsing. While hybrid blocks/text environment is a combination of textual and block programming which allows user to bring together the preferred features of textual and block programming.

Myriad kind of environments have been proposed to fulfil the unique demands and challenges in teaching introductory programming. From the survey did by Cheung et al (2009), they realize there is a gap for students in Grade 11-13 (junior secondary school). They run programming workshops to teach beginners programming aged from 8 to 18 years old (from elementary school to secondary school) using Scratch software. Students in secondary school feel too bored because of the environment simplicity and limitation. They stated that they prefer conventional textual programming environment. While for elementary, they are enjoying and respond best to the environment. The authors of this paper realize there is a gap for students in Grade 11-13 (junior high school). Students for that grade find that textual environments are too difficult while visual environments are too limited.

Lewis (2010) had done study on students' perception and learning outcome of block-based and text-based environment; Scratch and Logo. He found that small difference in performance between two groups who use Scratch and Logo. Scratch group have better ability in interpretation conditional statements. While, Logo group have better confidence to do programming. However, after the experiment done, both groups respond that programming course is difficult and hard to learn. Parson and Haden (2007) stated that block-based programming environment apparent as simple and far from real programming because fewer syntactic restriction and limitation in building complex programs. However, Weintrop (2015) found that 92% of students who joined his experiment have good respond in block-based programming. They view block-based programming is easier than text-based programming because blocks are closer to natural language and no syntax to remember and write.

Matsuzawa et al (2015) have developed hybrid programming environment for Java language to track either text or block that students prefer. Earlier in the experiment, he found that students prefer block-based modality. But, at the middle of experiment and towards the end, students moving to text-based. Block-based modality proven in helping novice to learn programming (Weintrop and Holbert, 2018). It leads by the development of block programming environment such Scratch, Alice, and Blockly as an introductory programming tools for novice or younger learners. While, there is a question of the suitability of such modality in transitioning students to future computer science learning. Weintrop and Holbert (2018) revealed that at early phase of their experiment, students choose block to start building their program, but over the time, they choose to use text modality, and returned to block to add new code that have not been used yet during their program development.

For some education level, block-based seems too simple but text-based seems too challenging for them. Thus, hybrid programming environment proof to be suitable environment for teaching introductory programming for students with minimal or no programming experience (Cheung, 2009). Plus, this environment increases their confidence and interest in programming. As stated in research by Bau et al (2017), hybrid or bidirectional mode switching provides two-ways transformation between text language and block mechanism. Dual-mode environment benefits users from

learnability of block mechanism and get the competency of text mechanism. In addition, it supports error handling where single mode of block does not support. User can learn programming by using block and in the same time, they also can experience error handling like traditional (text-based) programming language. Blanchard (2017) makes research of the impact of hybrid programming environment to computer science competency, confidence, and interest among students.

Programming environments used by students in most college or university are text-based environment such DevC++ and Eclipse (Li et al., 2016). This require students to manually type the codes to write a program. They need to remember and write the correct syntax, or the compilation of their program will be not successful. This become a problem to the beginners of the programming language. According to Lahtinen et al. (2005), the lack of understanding programming concept among beginners come from programming environment complexities and language syntax. Some difficulties that are faced by beginners are (Robins et al., 2003, Truong et al., 2007, Renumol et al., 2009):

i. Installing and setting class paths for compiler
ii. Learning functionalities of programming editors
iii. Understanding programming questions
iv. Writing code using programming language syntax knowledge
v. Describing the program logic and the difficulty of translating logic to program
vi. Poor quality of assistance offered by trainers
vii. Lack of useful information about library functions and header files
viii. Understanding compiler error messages
ix. Fix errors, as determined during the debugging process

Because of that, the choice of programming environment for beginners are important to make a significant difference in learning and to encourage them to do a programming. Usually, programming environments are developed to meet professional programmers' needs. Besides, this environment contains extensive sets of

concepts and features where hard for novices to understand especially in understanding error and warning message (Pears, et al., 2007).

Programming is difficult and requires many works including dedication and training. Difficult is not only to understand concept or course structure, but lack of motivation can lead to student frustration (Figueiredo and Garcia-Penalvo, 2018). To overcome this problem especially among youth, many researchers inspired to find ways to helping teachers in teaching programming, and students in learning programming. In addition, some researchers stimulated to doing some research in effectiveness of programmable robot or simulator robot as an aided-tool in introductory programming course (Major, 2014; Gonzalez & Valcarcel, 2017; Barr, 2011). Referring to Bau et al (2017), programming environment for novice should provide example that easy to find apply and installation free. Besides, vocabulary and grammar should be visible. It also should describe simple concepts by using clear words and high-level abstraction.

According to Liu, et al. (2013), robotic education used by many schools to engage students in science, technology, engineering, and mathematics (STEM) activities. In Malaysia, many schools use robotic technology as a tool in support teaching of problem solving (Tukimat, 2014). In placing more emphasis, Major (2014) claimed that assistant tool like robot simulation offered a more effective and enjoyment means in learning introductory programming. Gonzalez & Valcarcel (2017) stated that learning programming with robotic interference make learning process become meaningful and fun, through teamwork and collaboration.

Besides, the use of programmable robot in education is accepted and convinced by preschool teachers that joined teacher training day. Barr (2011) claimed that robotic education has positive impact to develop computational thinking and programming skills. Research done by Major (2014) stated that use of robot simulator supports effective learning programming to beginners. Based on experiment done by him, learning programming using robot simulator become a valuable and engaging approach to learner especially novice.

However, programming with robotic can become complex because of the increasing number of motors, sensors, and features of the robot to fulfil some objective. The complexity of robot makes the programming environment become complex and makes the end-user difficult to do programming on robot (Laval, 2018). Besides, Laval (2018) and Murphy (2014) claimed that 50% of robots' failure is because of human-robot interaction with non-adequate programming environment. According to Chown et al (2006), Tekkotsu environment (Touretzky et al, 2005) have complex environment which requires several tutorials to write a simple program. Simpler design should be developed for general use in classroom and have a low learning curve to make robot move. Low barrier to entry is important to minimize the time requirement to learn creating program (Cross, 2013).

1.3 Problem Statement

It is crucial to develop an environment specifically designed for the needs of beginning programmers to learn introductory programming. In the past few decades, the development of programming environment, tools and languages increased to support learning introductory programming processes. Various types of interventions have been used to overcome the problems in learning introductory programming and help students to develop programming skills. However, novice still find difficult in grasp the programming concept. Weintrop and Wilensky (2017) stated that modality affected learner in attitudes, perception and conceptual learning. There are three type of programming modality; text-based, block-based, and hybrid. The general research questions this research tries to answer:

"What programming environment that suitable for secondary school students to learn introductory programming through robotic?"

To answer this question, a set of research questions are defined as follow:

(i) Is a block-based, text-based or hybrid programming environment more suitable for secondary school students?

(ii) What is the strength and potential drawbacks to block-based, text-based and hybrid programming environment does secondary school students will face?

(iii) What is the most suitable programming environment are being used in introductory programming course?

(iv) How to evaluate the proposed environment for teaching introductory programming?

1.4 Research Aim and Objectives

The overall aim of this research is to propose programming environment that suitable for secondary school students to learn introductory programming. The aim of the research incorporates four objectives:

(i) To study the existing programming environment for teaching introductory programming

(ii) To study the students' perception on block-based and text-based programming environment

(iii) To propose a programming environment for teaching introductory programming

(iv) To evaluate the effectiveness of the proposed environment for teaching introductory programming

1.5 Research Scope

In this research, the scope of the study is defined as follows:

(i) This study focuses on the teaching introductory programming for secondary school students

(ii) Student age range between 13 and 17 years

(iii) This study focuses on the programming environment used for teaching programming

(iv) This study focuses on teaching programming through robotic

1.6 Significant of Study

Finding from this research will contribute to our understanding either block-based, text based or hybrid programming environment fits into more formal, structured educational spaces for secondary school. In addition, the intended audience for this paper not only among teachers and curriculum designer of secondary school. But, it includes computer science community who are planning, designing, and revising a new course to teach introductory programming to novices.

The practices, tools, and curriculum resulted from this research will become the standard for secondary schools especially schools in Malaysia. The lack of interest of programming subject among university students is because they struggle to understand the programming concept plus the programming environment itself make the problem worsen. In addition, they have not been practically taught of programming in their previous studies, for example, in a secondary school. Exposure to programming in early stage of education can change the perception of programming to better view. Furthermore, the use of right programming environment and tool will help students to learn programming in more effective way. Besides, learning programming subject will be one of the ways to improve problem solving and critical thinking skills. We are confident that the proposed environment is effective at teaching introductory programming for secondary school.

1.7 Structure of Thesis

Chapter 1 describes the overview of the study by explain the research background, problem statement, research aim, objective, scope and significant of study. The remainder of this thesis is broken down into six chapters. The first of this

REFERENCES

Aktunc, O. (2013). A Teaching Methodology for Introductory Programming Courses using Alice. *International Journal of Modern Engineering Research*, 3(1), 350-353.

Alexander Ruf, Andreas Mühling, Peter Hubwieser. (2014). Scratch vs. Karel – Impact on Learning Outcomes and Motivation. *WiPSCE 2014*.

Ardublock. (2016). Ardublock a Graphical Programming Language for Arduino. Available online: http://blog.ardublock.com/, accessed 11-19-2016.

Armoni, M., Meerbaum-Salant, O. & Ben-Ari, M. (2015). From Scratch to "real" programming. *ACM Transaction Computer Education*.

Azad A. and F. Kohun. (2009). Considerations for Selecting a Programming Language to Teach Perspective Teachers. *Alice Symposium*, Duke University, Durham, NC.

B. Kaucic and T. Asic. (2011) Improving introductory programming with Scratch?. *Procedure 34th International Convention MIPRO*, 1095–1100.

Barr, D., Harrison, J., y Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology*, 38(6), 20-23.

Bau, D., Anthony, D. & Mathew D. (2015). Pencil Code: Block Code for a Text World. *ACM IDC '15.* June 21 – 25, 2015.

Bau, D., Gray, J., Kelleher, C., Sheldon, J. & Turbak, F. (2017) 'Learnable programming: blocks and beyond'. *Commun. ACM*, 60(6) pp. 72–80

Ben, E., Cyr, M., and Rogers, C. (2013) Lego engineer and RoboLab: Teaching engineering with LabView from kindergarten to graduate school. *International Journal of Engineering Education,* 16(3):181-192. 12.

Bergin, J., Lister, R., Owens, B., & McNally, M. (2006) 'The first programming course: ideas to end the enrolment decline', *ACM SIGCSE Bulletin*, 38(3), pp301-302).

Blanchard, J. (2017). Hybrid Environments: A Bridge from Blocks to Text. ICER'17. August 18–20, 2017. Department of CISE University of Florida Gainesville, FL, USA, Tacoma, WA, USA

Briana B. Morrison, Betsy DiSalvo. (2014). Khan Academy Gamifies Computer Science. *Proceedings of the 45th ACM technical symposium on Computer science education,* 39-44.

Bruckman, A. and Edwards, E. (1999). Should we leverage natural-language knowledge?. *Procedure of the SIGCHI conference*, 207–214.

Caballero-González, Y. A. & García-Valcárcel, A. G (2017) Development of computational thinking and collaborative learning in kindergarten using programmable educational robots: a teacher training experience. TEEM 2017, October18 – 20 2017. Cádiz, Spain.

Carle, A. & Schertle, R. (2017) mBot for makers: Conceive, Construct, and Code Your Own Robots at Home or in the Classroom.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., and Hadfield, S. M. (2005). RAPTOR: a visual programming environment for teaching algorithmic problem solving. *Technical Symposium on Computer Science Education*. 176-180.

Chown, E., Foil, G., Work, H., and Zhuang, Y. (2006). AiboConnnect: A simple programming environment for robotics. *Proceedings of the Nineteenth*

International Florida Artificial Intelligence Research Society Conference, Melbourne Beach, Florida, USA, May 11-13, 2006

Cooper, S., Dann, W., and Pausch, R. (2003). Teaching objects-first in introductory computer science. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education,* 91-195.

Corral, J.M.R, Balcells, A.C., Estévez, A.M., Moreno, G.J., Ramos, M.J.F. A game-based approach to the teaching of object-oriented programming languages. *Computer and Education*, vol. 73, 83-92.

Cross, J., Bartley, C., Hamner, E., Nourbabkhsh, I. (2013). A Visual Robot-Programming Environment for Multidisciplinary Education. *IEEE International Conference on Robotics and Automation (ICRA),* Germany, May 6-10, 2013

D. Goulet and D. Slater. (2009). Alice and the introductory programming course: An invitation to dialogue. *Information Systems Education Journal*, Vol. 7.

Denny, P., Luxton-Reilly, A., Tempero, E. and Hendrickx, J. (2011). Understanding the syntax barrier for novices. *Procedure of the 16th Annual ITiCSE,* 208–212.

Duncan, C., Bell, T., and Tanimoto, S. (2014). Should your 8-year-old learn coding?. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education,* 60-69

Figueiredo, J. & García-Peñalvo, F. J. (2018). Building Skills in Introductory Programming. In Proceedings of the 6th International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM 2018), October 24-26, 2018. Salamanca, Spain, New York, NY, USA.

Grout, V., and Houlden, N. (2014). Taking computer science and programming into schools: The Glyndwr/BCS Turing Project. *Procedia – Social and Behavioral Sciences*, 141(25), 680–685.

Hunpatin, O., O'Hare, C., Thomas, R., Brylow, D. (2016). A Browser-based IDE for the MUzECS Platform. *The 22nd International Conference on Distributed Multimedia Systems.*

Jawawi, D., Mamat, R., Ridzuan, F., Khatibsyarbini, M., Zaki, M. (2015). Introducing Computer Programming to Secondary School Students using Mobile Robots. *10th Asian Control Conference 2015 "Emerging Control Techniques for a sustainable World".* May 31 – June 3. Kota Kinabalu, Malaysia.

Joey. C. Y. Cheung, Grace Ngai, Stephen C. F. Chan and Winnie W.Y. Lau. (2009). Filling the Gap in Programming Instruction: A Text-enhanced Graphical Programming Environment for Junior High Students. *Proceedings of the 40th ACM technical symposium on Computer science education,* 276-280.

Jones, D. W. (2004) Introduction to System Software. The university of IOWA Department of Computer Science.

Jost, B., Ketterl, M., Budde, R., & Leimbach, T. (2014). Graphical Programming Environments for Educational Robots: Open Roberta - Yet another One? *IEEE International Symposium on Multimedia.*

Kay, J. S. (2003). Teaching robotics from a computer science perspective. *Journal of Computing in Small Colleges*. 19(2), 329-336.

Kelleher, C., and Pausch, R. (2005). Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2), 88-137.

Lahtinen, E., Ala-Mutka, K., and Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14-18.

Laval, J. (2018). End User Live Programming Environment for Robotics. *Robotics and Automation Engineering Journal*, 3(2), 14-18.

Lewis, C. M. (2010). How Programming Environment Shapes Perception, Learning and Goals: Logo vs. Scratch. *ACM SIGCSE,* pp346-350.

Li, H. F., Liang, T. Y., Peng, H. T., (2016). A Block-Oriented C Programming Environment. *International Conference on Applied System Innovation (ICASI).*

Liu, L., Zhang, J. X., Ordóñez de Pablos, P., and She, J. (2014). The auxiliary role of information technology in teaching: Enhancing programming course using Alice. *International Journal of Engineering Education*, 30(3), 560–565.

M. Bajzek, H. Bort, O. Hunpatin, L. Mivshek, T. Much, C. O'Hare, D. Brylow. (2016). MUzECS: Embedded Blocks for Exploring Computer Science. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education,* 127-132.

Major, L. (2014). *An Empirical Investigation into The Effectiveness of A Robot Simulator As A Tool To Support The Learning Of Introductory Programming.* PhD Thesis. Keele University.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computer Education.*

Mannila, L., Peltomaki, M., and Salakoski, T. (2006). What About a Simple Language? Analyzing the Difficulties in Learning to Program. *Computer Science Education*, 16(3), 211-227.

Matsuzawa, Y. et al. (2015). Language Migration in non-CS Introductory Programming through Mutual Language Translation Environment. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 185–190.

Meerbaum-Salant, O. (2011). Habits of programming in scratch. *In ITiCSE '11*, 168–172.

Mladenovic, Monika & Krpan, Divna & Mladenovic, Sasa. (2017). Learning programming from Scratch.

Murphy, R. R. (2014) Disaster Robotics. The MIT Press

Parsons, D. & Haden, P. (2007). Programming Osmosis: Knowledge Transfer from Imperative to Visual Programming Environments. *20th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2007)*.

Parsons, D., and Haden, P. (2007). Programming Osmosis: Knowledge Transfer from Imperative to Visual Programming Environments. *In S. Mann & N. Bridgeman (Eds.), Proceedings of The Twentieth Annual NACCQ Conference,* 209-215.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 39(4), December 2007.

Pendergast, M. (2006). Teaching introductory programming to IS students: Java problems and pitfalls. *Journal of Information Technology Education*, vol. 5, 491–515.

Powers, K., Ecott, S. and Hirshfield, L. (2007). Through the looking glass: teaching CS0 with Alice. *Proceedings of the 38th SIGCSE technical symposium on Computer Science Education.*

Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., Proulx, V. and Carlisle, M. (2006). Tools for teaching introductory programming: what works?. *ACM SIGCSE Bulletin*, 38(1), 560-561.

Renumol, V. G., Jayaprakash, S. & Janakiram, D. (2009). Classification of cognitive difficulties of students to learn computer programming. *Indian Institute. of Technology*, Department of Computer Science, Chennai.

Ridzuan, F. (2016). Comparative Study on Teaching Programming Tools for Secondary School Students. Universiti Teknologi Malaysia: *Degree Thesis.*

Robins, A., Rountree, J., and Rountree, J. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 132-172.

Robson, C. (2011) *Real World Research.* 3rd Edition. John Wiley & Sons.

Stefik, A. and Siebert, S. (2013). An Empirical Investigation into Programming Language Syntax. *ACM Transactions on Computing Education.* 13(4), 1–40.

Stutterheim, J., Swierstra, W., and Swierstra, D. (2013). Forty hours of declarative programming: Teaching Prolog at the Junior College Utrecht. *Electronic Proceedings in Theoretical Computer Science EPTCS Electron. Procedure Theory Computer Science,* 50-62.

Taheri, S. M., Sasaki, M., and Ngetha, H. T. (2015). Evaluating the effectiveness of problem-solving techniques and tools in programming. *Science and Information Conference (SAI)*, 928-932.

Touretzky, D. S. and Tira-Thompson, E. J. (2005). Tekkotsu: A framework for AIBO cognitive robotics. *Proceedings of the Twentieth National Conference on Aritificial Intelligence*. (AAAI-05).

Truong, N. (2007). *A web-based programming environment for Novice Programmers.* Ph.D. dissertation Faculty of Inform. Technology, Queensland University of Technology, Queensland.

Tukimat, N. M. N. (2014). Perisian Membantu Pembelajaran Pengatucaraan dengan RoboKar. Universiti Teknologi Malaysia: *Degree Thesis.*

Vihavainen, A., Miller, C. S., and Settle, A. (2015). Benefits of Self-explanation in Introductory Programming. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 284- 289.

Weintrop, D. & Wilensky, U. (2017). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Trans. Comput. Educ.* 18(1) Article 3, October, 2017, 25 pages.

Weintrop, D. (2015). Blocks, Text, and the Space Between the Role of Representations in Novice Programming Environments. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).*

Weintrop, D. and Wilensky, U. (2015). To Block or not to Block, that is the Question: Students' Perceptions of Blocks-based Programming. *Procedure of the 14th Annual IDC Conference*.

Weintrop, D. and Wilensky, U. (2015). Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. *ICER '15*.

Wellman, B. L., Anderson, M., and Vrbsky, S. V. (2009) 'PREOP as a tool to increase student retention in CS'. *Journal of Computing Sciences in Colleges*, 25(2), pp167-175.

Wing, J. (2006) Computational Thinking. Communications of the ACM. June 27-30, 2016. Standford, CA, USA, pp33 – 35

Yin, R.K., (1984). Case Study Research: Design and Methods. Beverly Hills, Calif: Sage Publications.

Website

Berkeley Logo. (2016). https://people.eecs.berkeley.edu/~bh/logo.html, accessed 30-06-2016.

Google Blockly. (2016). https://developers.google.com/blockly/, accessed 30-06-2016.

Heroku. (2018).

https://www.heroku.com/, accessed 28-12-2018.

Karel Programming Language. (2016). https://en.wikipedia.org/wiki/Karel_(programming_language), accessed 11-19-2016.

Malay Mail Newspaper. (2016)

RobotC, 2018(www.robotc.net)

Schor, J. (2016). Co-Founder and CEO of CodeMonkey Studios. Available online: https://www.fractuslearning.com/2016/02/15/teaching-kids-code-text-based-vs-block-based/, accessed 11-19-2016.