Using multimodality to teach introductory programming

Mufhulufheli Mabilo
University of Cape Town
MBLMUF001@myuct.ac.za

Abstract

Most High School in South Africa do not offer Information Technology related subjects in their curriculum, which has challenges for first year's students who are enrolled in Computer Science. They do not maintain their engagement with the work as it is difficult to understand. The goal of this paper is two-structure, structuring reviews of related literature with regards to learning programming and learning system that incorporates multimodality to help first year students understand programming code. Based on the findings in literature, there was a gap with regards to understanding the basic concepts of programming code which includes syntax. This review will examine such work by focusing on a learning system that allows students to step through the code with the help of speech tools and subtitles will help students understand programming code basic concepts.

1 Introduction

Computer Science courses require good abilities from a student. It requires a student to understand syntax and semantics of any desired programming language and use their knowledge and creativity to solve problems and tasks. Computer Science is a result of a combination of logical thinking with creativity.Challenges regarding students learning programming language include [14].

- Learning a new programming language.
- Not sure of how or where to start with a given task.
- Ability to be able to debug.
- Not understanding the task given and spending a lot of time with no output.
- Lack of asking for help and being specific of the problem.
- Lack of consistency with regards to practicing coding.
- Lack of problem solving and breaking problems to sub-problems.

The most important way to solve these challenges is through practice over time. However with these challenges students drop-out or change courses within their first year, as they do not have support and motivation to continue with programming.
An introduction programming language must suit the following requirements:

- Simplicity.
- Consistency.
- Typicality.
- Usability.
- Flexibility.
- Good documentation.
- Good IDE.
- Most importantly easy to setup.

With the requirements it will help students understand the content [33].

2 Computer Science

2.1 What is Computer Science

Computer Science is defined to be the study of computer hardware and software, with the development of software that it is used in computer hardware to function together with the computer hardware [27].
Programming can be defined as the process of converting a mental plan or idea into a computer program. With the main goal being to develop skills to create computer programs that solve efficiently real problems [7].

2.2 Courses for programming for first years

Learning to program for most is difficult for the first time and potentially many end up dropping out. Some problems may include inability to decompose a problem into sub-problems, lack of being able to read and understand your code, Understanding coding principles and synthesizing new knowledge. The fundamental learning's can include loops, variables, recursion and variable parsing. Most importantly students must be taught in a way that they don't have issue with or confuse learning programming with the learning of programming language [25].
Factors that contribute to challenges with programming include, the increase of students diversity and poor attendance, with poor teaching methodologies[13]

2.3 Learning's that are considered when learning Computer Science

2.3.1 Behaviourism

The view of how students approach the course is always based on well the lecture treats the students[27]

2.3.2 Cognitive

It includes the ability of a student to logically think through information when they are learning. Have students think around about the information they are being taught [31].

2.3.3 Constructivism

It involves the combination of new ideas to existing ideas with the understanding of the subject, and through that conclusions are developed [17]. The attention to all students should be taken into consideration.

2.3.4 Design

Design allows to construct a goal of finding a way to have a way to have a balance between theoretical knowledge and how the theoretical knowledge is used to teach [28].

2.3.5 Humanism

Humanism allows the development of creativity and interest engagement in programming [35].

2.4 Role of fun in learning programming

Learning environment must at least be fun and enjoyable towards students, as fun plays an important role towards students.

2.4.1 Emotions, learning and attitudes

Learning something can be supported by ability of self report, oral response, written response and direct observations [30].

2.4.2 Conceptualizing fun

Teaching through fun has allowed all age groups to learn faster and better. Learning with lack of interesting information has showed that people lose interest of what they're learning [20].

3 Learning the basic principles of programming

3.1 Basic programming learning support systems

A literature review of the International Conference on Computer Systems and Technologies showed that they have been many tools that help solving the challenges of learning programming, with many of this tools using animation and simulation techniques, with the help of taking into consideration the human visual system potential. Animation by far contributes to a clear and better understanding with comparison with static formats. There are visualisation systems that are available which includes:

- There are systems that focus on algorithms and complete programs.
- There are systems that focus on low level details, that display data structures and their growth during a program.
- There are systems that focus on high detail level, displaying behaviour of program, component relation and methodologies.
- Some systems animate predefined programs and data structures.
- Some systems accept students programs and allows them to see they're work and allows them to make necessary corrections.
- Some systems like MRUDs(Multiple Representation for Understanding Data Structures) [12], have used multiple visual representations to demonstrate linear data structures such as tables, stacks, queues, and lists, with these representations suitable for students with less or no knowledge of data structures.
- Some systems were proposed using either representation or algorithm animation which include , BALSAII [3] , VIP [21] , Xtango [34], Jeliot 2000 [16], Trackla[15].

With the use of knowledge and understanding, new programming languages were developed, including:

- Iconic programming language which include BACII [4] and BlueJ [14].
- Design languages which include G2 [10].
- Web-based environments which include Jhavé [23].

Also techniques of artificial intelligence that support programming teaching and learning were developed which were mainly tutoring systems which includes Lisp-Tutor [2], C-Tutor [33] and Cimel ITS [22].

3.2 Problem associated with the learning support systems

The support systems were designed to support academia with regards to supporting programming teaching and learning. With the acknowledgment that there are many systems designed to support programming learning, there is still a continuation of reports for difficulty in learning programming. With the answers being that the systems may not be well known by teachers and learners, and some systems are not well suited for students that need support with regards to learning programming. However they are factors that results in student difficulties:

- Insufficient learning study methods..
- Students difficulty in solving problems.

With the factors mentioned, the major reason behind students difficulty in learning programming is low problem solving capabilities, with regards to every basic problem as programming requires critical thinking and creative thinking, therefore an improve in problem solving would benefit students with understanding programming.

3.3 Characteristics essential for the development of basic programming learning support systems

3.3.1 Taking into consideration preferred learning styles

It influences the type of action that will be presented to students for learning. It requires knowledge about learning styles.

3.3.2 Using patterns of programming

When students are unable to solve a particular problem or a part of the problem, they should be presented with incomplete programs with necessary components for solving the program and it allows students to write programs not from scratch, but it basically allows them to build from existing problems, with a better understanding of the program.

3.3.3 The use of games

The idea with the use of games is to develop generic problem solving abilities, through attractive and stimulant activities, with the main reason being attracting students to the learning environment and an engaging learning environment [5].

4 Using robots for introductory programming

A literature review at the Aubun University [8] have developed a secondary component to an introductory programming course that focuses on problem solving. They made use of LEGO mind storms robot kits as an added application towards problem solving techniques. Students learn to write simple and easy programs for robots, so they can solve programming problems they're given.
The benefits of programming a robot includes:

- The visuals, aural and kinetic feedback of a robot is found to make students engage in the course, rather than a simple text based program.
- The interest towards robots allows students to spend more time on the program [9].
- It is easier for students test and debug their programs.
- Making use of the robots allowed students to be introduced to real world programs [29].
- Lastly it allowed students to think of the robot as an object, which is a great benefit towards understanding object-oriented programming.

However with the benefits, there was a major disadvantage which was students have limited access to the robot, as the robot was expensive [32].
However the problem was solved through developing a web interface that allows students to remotely program the robot and review the results via webcam. Teams were able to refine their solution within class meetings [8].

5 Using gaming for introductory programming

A game can be described as a system based on formal rules, with quantifiable and variable outcomes, the outcomes are linked to different values and the player feels attached and cares mostly about the outcome [26]. Games have features present:

- They have rules.
- They have a win stage.
- They have outcomes and feedback.
- They require critical thinking and problem solving.
- They follow a form of a play.
- They are adaptive.
- There is social interaction within a game.

Every game is based on the fact that it provides feedback which makes players to be committed as feedback is in the form of score or levels [11].

5.1 Using board game for introductory programming

Board games includes board, card and dice game. Many computer scientists have a natural affinity for board games, and most board games have similar trait with computer programming, with few rules giving rise to deep and fascinating complexities [6].

5.2 Gaming for introductory programming

Everyone tends to find gaming to be massively engaging and most people are motivated to keep playing [37]. The engagement is of great benefit as it allows students to be more engaged in the coursework.
Game based learning systems gives a visual medium in which to apply practical and puzzle based problem solving. The visual component of these learning systems has been notified to help interpret abstract concepts and helps with student engagement [24].
The game based systems are found to be fun and enjoyable, hence learning through having fun leads to a higher degree of effort by students with regards to practicing the coursework and a positive approach towards the course.
With regards to student cognition and learning systems, students using game based learning systems have the freedom of interaction with regards to approaching the game and the content within their on pace and desire. Which is important as it allows students who are learning programming to use their own learning styles and adjust the pacing and emphasis of what they are learning for a better understanding [36].
Some game based learning systems are said to be effective tool for learning introductory programming. They are many educational games, but there is lack of evidence of their impact on academics, as games vary and the games can have a no effectiveness on some students[37].

6 Using ADRI model system for introductory programming

The ADRI (Approach, Deployment, Result, and Improvement) [19] model is an analytic tool, it is a quality assurance model for self review and external review. It was developed from the Plan-To-Do-Check-Act model. They are four approaches of ADRI model:

- **Approach-problem solving strategies** - It consists of the development of goals, strategies and plans with regards to learning programming.
- **Deployment** - programming knowledge - The steps for achieving goals must be clear, proper planning must in place to ensure there's sufficient programming knowledge .
- **Results** - Shows the process to solve problem statements and helps with understanding programming in a different way.
- **Improvement** - Recommendations for students with regards to

The four stages helps new students to understand basic programming in a simpler way. It also helps with balancing problem solving strategies and programming knowledge [19].

7 Mobile application development for introductory programming

Every University student has a smart phone around University campus, as they spend more time on their phones. Introducing mobile application development at an introductory level would be of great use, as they would be engaging [18].

Students can be able to test their work through actual devices or emulators, which gives an advantage to be able to debug and test their code quicker and easy, as they know the desired outcome.
With the mobile platform for which applications are being developed there's a great of planning and consideration that students have to undergo. However mobile applications have characteristics that differ from desktop platforms which include:

- Bandwidth .
- Processor.
- Speed.
- Screen size and resolution.
- Memory consumption.
- battery life.
- user input.

Hence with the characteristics students learns how to be mindful of how mobile applications handles memory for instance, and would be able to apply the knowledge to other fields.

Computer Science students with experience in mobile application development can be in native or cross platform, can use the knowledge they have gained through mobile application development and use the knowledge in traditional development which gives an advantage of creating more flexible students developer and allows them to learn faster with better understanding[18].

8 Conclusion

This survey has examined a larger body of literature related to tools for helping teach introductory programming. The research shows the importance of learning computer science and programming, which includes developing a better critical and creative approach to real life problems in the software industry. With the main reason of developing this skills for developing skills for creating computer programs to solve real life problems. There are however challenges with learning programming which includes, the increase of poor attendance of classes and activities, with poor teaching methodologies and lack of motivation to learn programming as learning programming is considered to be difficult to understand [1].
Using robots to teach introductory programming showed a great effectiveness, as it allows students to learn programming more simpler as they learn to write simple and easy programs for robots, so they can solve programming problems they're given easier and faster, with a better understanding[8].
Using games to teach introductory programming however does not show enough effectiveness as games may vary and some games are not created for everyone specially.
The ADRI based approach promotes problem solving strategies, programming knowledge demonstration with a programming example and problem, which helps shift the understanding of programming to a different way that easier to understand [19].
The most effective approach to help create a system for teaching introductory programming, is to have a system that will allow students to have fun while learning, as learning through fun allows students to be more engaged with the coursework.

References

[1] Ozgur Aktunc. 2013. A teaching methodology for introductory programming courses using Alice. *International Journal of Modern Engineering Research (IJMER)* 3, 1 (2013), 350–353.

[2] John R Anderson. 1996. ACT: A simple theory of complex cognition. *American psychologist* 51, 4 (1996), 355.

[3] Marc H. Brown. 1988. Exploring algorithms using Balsa-II. *Computer* 21, 5 (1988), 14–36.

[4] Ben Anthony Calloni. 1992. *BACCII: An iconic, syntax-directed windows system for teaching procedural programming.* Ph. D. Dissertation.

[5] Carla Delgado, José Antonio Moreira Xexeo, Izabel F SOUZA, Marcio Campos, and Clevi Elena Rapkiewicz. 2004. Uma abordagem pedagógica para a iniciação ao estudo de algoritmos. In *XII Workshop de Educação em Computação.*

[6] Peter Drake and Kelvin Sung. 2011. Teaching introductory programming with popular board games. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. 619–624.

[7] José Figueiredo and Francisco García-Peñalvo. 2021. Teaching and Learning Tools for Introductory Programming in University Courses. In *2021 International Symposium on Computers in Education (SIIE)*. 1–6. https://doi.org/10.1109/SIIE53363.2021.9583623

[8] Aaron Garrett and David Thornton. 2005. A web-based programming environment for lego mindstorms robots. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*. 349–350.

[9] Michael Goldweber, Clare Congdon, Barry Fagin, Deborah Hwang, and Frank Klassner. 2001. The use of robots in the undergraduate curriculum: experience reports. In *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*. 404–405.

[10] Anabela Gomes and António José Mendes. 2007. An environment to improve programming education. In *Proceedings of the 2007 international conference on Computer systems and technologies*. 1–6.

[11] Juho Hamari, Jonna Koivisto, and Harri Sarsa. 2014. Does gamification work?–a literature review of empirical studies on gamification. In *2014 47th Hawaii international conference on system sciences*. Ieee, 3025–3034.

[12] Biffah Hanciles, Venky Shankararaman, and Jose Munoz. 1997. Multiple representation for understanding data structures. *Computers & Education* 29, 1 (1997), 1–11.

[13] Bassey Isong. 2014. A Methodology for Teaching Computer Programming: first year students' perspective. *International journal of modern education and computer science* 6, 9 (2014), 15.

[14] Michael Kölling, Bruce Quig, Andrew Patterson, and John Rosenberg. 2003. The BlueJ system and its pedagogy. *Computer Science Education* 13, 4 (2003), 249–268.

[15] Ari Korhonen, Lauri Malmi, and Panu Silvasti. 2003. TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises. In *Kolin Kolistelut-Koli Calling, Oct. 3-5, 2003 in Koli Finland*. University of Joensuu and University of Helsinki, 48–56.

[16] Ronit Ben-Bassat Levy, Mordechai Ben-Ari, and Pekka A Uronen. 2003. The Jeliot 2000 program animation system. *Computers & Education* 40, 1 (2003), 1–15.

[17] Philip Machanick. 2007. A social construction approach to computer science education. *Computer Science Education* 17, 1 (2007), 1–20.

[18] Qusay H. Mahmoud and Pawel Popowicz. 2010. A mobile application development approach to teaching introductory programming. In *2010 IEEE Frontiers in Education Conference (FIE)*. T4F–1–T4F–6. https://doi.org/10.1109/FIE.2010.5673608

[19] Sohail Iqbal Malik and Jo Coldwell-Neilson. 2017. A model for teaching an introductory programming course using ADRI. *Education and Information Technologies* 22 (2017), 1089–1120.

[20] IC McManus, Adrian Furnham, et al. 2010. "Fun, fun, fun": Types of fun, attitudes to fun, and their relation to personality and biographical factors. *Psychology* 1, 03 (2010), 159.

[21] A Mendes and Teresa Mendes. 1988. VIP-A tool to VIsualize Programming examples. *Proc. of the EACT* (1988).

[22] Sally H Moritz, Fang Wei, Shahida M Parvez, and Glenn D Blank. 2005. From objects-first to design-first with multimedia and intelligent tutoring. *ACM SIGCSE Bulletin* 37, 3 (2005), 99–103.

[23] Thomas L Naps. 2005. Jhavé: Supporting algorithm visualization. *IEEE Computer Graphics and Applications* 25, 5 (2005), 49–55.

[24] Thomas L Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, et al. 2002. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education*. 131–152.

[25] Sasha Nikolic, Montserrat Ros, and David B Hastie. 2018. Teaching programming in common first year engineering: discipline insights applying a flipped learning problem-solving approach. *Australasian Journal of Engineering Education* 23, 1 (2018), 3–14.

[26] Marc Prensky. 2001. Fun, play and games: What makes games engaging. *Digital game-based learning* 5, 1 (2001), 5–31.

[27] Jesús Ubaldo Quevedo-Torrero. 2009. Learning Theories in Computer Science Education. In *2009 Sixth International Conference on Information Technology: New Generations*. 1634–1635. https://doi.org/10.1109/ITNG.2009.294

[28] Armida Salazar. 2019. Design-Based Approach Learning Model for Teaching Computer Programming. https://doi.org/10.13140/RG.2.2.11062.11842

[29] Jerry Schumacher, Don Welch, and David Raymond. 2001. Teaching introductory programming, problem solving and information technology with robots at West Point. In *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No. 01CH37193)*, Vol. 2. IEEE, F1B–2.

[30] Dale H Schunk. 2012. *Learning theories an educational perspective*. Pearson Education, Inc.

[31] Dale Shaffer, Wendy Doube, and Juhani Tuovinen. 2003. Applying Cognitive load theory to computer science education.. In *PPIG*, Vol. 1. Citeseer, 333–346.

[32] Roland Siegwart and Patrick Saucy. 1999. Interacting mobile robots on the web. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[33] JS Song, SH Hahn, KY Tak, and JinHyung Kim. 1997. An intelligent tutoring system for introductory C language course. *Computers & Education* 28, 2 (1997), 93–102.

[34] John Stasko. 1992. Animating algorithms with XTANGO. *ACM SIGACT News* 23, 2 (1992), 67–71.

[35] Estelle Taylor, Marnus Breed, Ilette Hauman, and Armando Homann. 2013. Choosing Learning Methods Suitable for Teaching and Learning in Computer Science. *International Association for Development of the Information Society* (2013).

[36] Lynda Thomas, Mark Ratcliffe, John Woodbury, and Emma Jarman. 2002. Learning styles and performance in the introductory programming sequence. *ACM SIGCSE Bulletin* 34, 1 (2002), 33–37.

[37] Michael F Young, Stephen Slota, Andrew B Cutter, Gerard Jalette, Greg Mullin, Benedict Lai, Zeus Simeoni, Matthew Tran, and Mariya Yukhymenko. 2012. Our princess is in another castle: A review of trends in serious gaming for education. *Review of educational research* 82, 1 (2012), 61–89.