ROBOPROG: LEARNING OF FLOWCHARTS THROUGH A GAMIFIED EXPERIENCE

Reginald G Govender
University of KwaZulu-Natal
E-mail: govenderR4@ukzn.ac.za
ORCID: 0000-0002-3143-4050

Desmond W Govender
University of KwaZulu-Natal
E-mail: govenderD50@ukzn.ac.za
ORCID: 0000-0002-6115-9635

─Abstract ─

For the novice, learning their first programming language is perceived as a series of challenges. The core skill required is problem-solving, starting with the design of flowcharts. This paper presents RobotProg, a programming instructional tool that uses flowcharts to create a series of robotic actions through gameplay, creating a learning path through gamification using levels and missions. The aim is to provide a reflective overview of the gamified flowchart-learning tool, RobotProg, as a form of instruction. RobotProg flowcharts are executed by the built-in interpreter, which displays the movement of the robot on the computer screen. The tool offers pre-exposure to programming concepts prior to language-specific syntax and semantics, providing a seamless transition from simple logic of flowcharts to more complex programming languages in text-based environments later on. The contemplation of RobotProg through the lens of Fogg's Behavioural Model effectively addresses the factors of motivation, ability and trigger all at the same time. This results in a programmer who is an advanced beginner and efficient problem solver. RobotProg indirectly through human-computer interaction, develops an individual's algorithmic thinking while being mentally stimulating, building interest and curiosity.

Key Words: Coding, Flowcharts, Gamification, Robots

JEL Classification: C63, C79, C88, I23, Q55

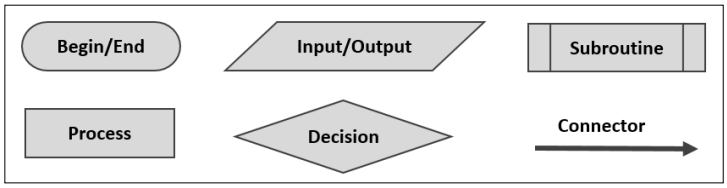
1. INTRODUCTION AND PURPOSE

1.1 Flowcharts

An early stage programmer (ESP) traditionally starts with the development of algorithms, by means of flowcharts, in a programming introductory course, before moving onto a specific programming language (Shneiderman, Mayer, McKay & Heller, 1977). Flowcharts offer a visual pictorial representation of a solution, thus allowing the visual ESP to exercise and develop their problem-solving skills through the selection of flow blocks, which describe the workflow of the solution. Flowcharts alleviate the stress of focusing on the syntactic details of programming languages and gives the ESP a chance to focus on acquiring problem-solving skills. Flowcharts depict an algorithm to solve a problem through the use of specific symbols, lines and arrows contained within a 'begin and end' of the flow process. This visual algorithm is hand drawn by the student. Seminal work conducted by Scanlan (1989) found that students prefer flowcharts over pseudocode for comprehending the algorithms. Flowcharts allow for a conceptual representation of input, process, output, assignments, conditions and loops.

Studies have shown that flowcharts have proven to be effective in the writing and understanding of algorithms, allowing the student to create a visual process (Ray, Herrera-Cámara, Runyon & Hammond, 2019; Cabo, 2018). However, a study conducted by Shneiderman et al. (1977), reported that flowcharts had no benefit for ESPs in the comprehension and debugging of computer programs. It was found that students tend to avoid using flowcharts, instead they start coding without a complete understanding of the problem, creating an unsatisfactory solution (Gul, Asif, Ahmad & Ahmad, 2017). The use of flowcharts for a novice can be overwhelming and tedious, coupled with rules, such as the need to always draw a terminal symbol to begin and end a flow. In addition, one must know when to use each symbol between the 'begin and end'. As remarked by Singh (2020) flowcharts are "time taking way and might become arduous …redrawing a flowchart for integrating changes/modifications it might sound a boring task" (p. 17). This makes little or no sense to the ESP let alone all the different symbols that serve unique purposes. The most common symbols are: begin, end, input, output, processing and decision (Figure 1).

Figure-1: Some common flowchart symbols



Source: Adapted from Singh: 2020:16, Turner: 2006: 96, Myler: 1998: 32-36, Chapin: 1970: 122-123)

1.2 Games, economics and human engagement

The degree of classification of human learning can be rated by Bloom's cognitive taxonomy. The syntactical structures required by high-level text-based programming languages involves memorising and rote learning of language syntax. This aligns with activities that are at Bloom's lowest level of human learning. Flowcharts can provide a visual process with minimal language syntax while one works at a conceptual level. This is rated to be of a higher level of human learning activity. Therefore, the learning of flowcharts is crucial to the development of the ESP and should not be overlooked as being a time waste.

Gamification is a design approach that uses a gameplay design in various contexts for inducing learning experiences (Buckley, 2016) in a form of a game that supports different activities and behaviours (Huotari & Hamari, 2017). The design of programs based on gamification is used in many contexts and have proven to be beneficial in the learning process (Huotari & Hamari, 2017). Morris (2018), reported that the total consumer spending on games was up 40% in the first half of 2018 in the United States of America. In the South African context gaming grew from R29.7-million to a R100-million industry within a period of two years marking more than a 50% increase and it continues to grow (Tarentaal, 2019; Collins, 2017). On a global platform, the gaming revenue is set to reach a soaring 100-billion US dollars by 2021 making up 59% of revenues in the global market (Wijman, 2018). Living in the Digital Age with the unfolding of the fourth industrial revolution with so many technological advancements, many have found leisure in digital gaming, thus serving as an ideal platform to integrate learning. The use of gamification in learning has proven to be successful in students' understanding of computing principles and programming (Lo & Hew, 2018), as it is a technological development that involves a high level of human engagement.

Many studies have been conducted that investigate the use of flowcharts for various purposes (Cabo, 2018; Chen & Morris, 2005; Lehman, 2000; Shneiderman et al., 1977; Knuth, 1963). However, none of these studies examine flowcharts with respect to gamification and the possible benefits to the novice programmer. In addition, the correctness of a flowchart has many possibilities allowing for multiple solutions to a problem. Along similar lines, Singh (2020) states "there are no standards to find out the correct extent of details that should be incorporated in making a flowchart" (p.17).

1.3 Scholarly significance of the work

The dawning of the fourth industrial revolution has placed an enormous priority on coding and robotics. These skills are crucial in preparing the current generation for the future workplace 2030. There is a need for programmers to possess problem-solving skills, apart from the actual development of programs. It is important to guide student's learning of programming in more effective and efficient ways. The learning of flowcharts based on game design with robots is ideal to develop the ESPs coding skill. Unlike the use of block-based programming that offers a farfetched experience from programming in a text-based environment, RobotProg incorporates the animated environment in gameplay through the use of flowcharts. RobotProg is an educational tool that supports and prepares the ESP to become a problem solver, through the use of flowcharts. The instructional tool teaches the individual indirectly through gameplay, to create algorithms that allow the robot to complete each mission and proceed to the next level. RobotProg was not evaluated with students as this was not the aim of this paper.

2. THEORETICAL PERSPECTIVE

Flowcharts contain specific symbols namely: terminal input/output, process, decision, module call and connectors, which are integrated to create a visual description of an algorithm - proposing a solution. RobotProg must not be confused with block-based programming environments that are notably offered through Scratch. Flowcharts form the foundation of activating an individual to start thinking algorithmically. Algorithmic thinking encompasses the process of following step-by-step instructions to solve a problem (Hsu & Wang, 2018; Burton, 2010; Futschek & Moschitz, 2010). It is of importance to engage a shift towards the desired behaviour, given that flowcharts are sometimes overlooked as a topic commonly offered in introductory programming courses, because they are perceived to be boring and dreary. For this reason, Fogg's Behaviour Model is utilised as it analyses factors that can produce or lead to a certain desired behaviour. Fogg's Behaviour Model is essential to human-computer interaction (HCI), especially to environments that involve gameplay with the intent of providing education instructions, aimed to create a meaningful learning experience. This behavioural model is helpful in the analysis and design of persuasive technologies (Fogg, 2009). Fogg's Behaviour Model is a product of three factors: motivation, ability and trigger, which simultaneously acquire the target behaviour (Fogg, 2009), which in this case is gearing and developing a programmer's mindset through the use of flowcharts. According to Fogg's

Behaviour Model: *motivation* is the core of the learning experience. For example, if a student can solve a task but is not motivated, they would not do so or would solve the task with reluctance. *Ability* is another factor that contributes to behaviour. If one is motivated but does not have the ability, then the desired behaviour cannot be achieved. However if one is highly motivated, one can find the means to complete the task, therefore gaining the ability. *Motivation* and *ability* are still not enough for an individual to achieve the desired behaviour. There needs to be a *trigger*. The *trigger* is connected to *motivation,* as it creates the spark that motivates the individual (Fogg, 2009).

3. METHODS, TECHNIQUES, OR MODES OF INQUIRY

The method of software evaluation carried out was *tutorial-based evaluation* informed by *user perspective*. *Tutorial-based evaluation* was created by the Software Sustainability Institute and is founded on subjective experiences (Jackson, Crouch & Baxter, 2011). RobotProg employs a procedural programming technique. Upon opening RobotProg, the user is presented with two windows, a list of flowchart symbols available at level one; and a blank flowchart canvas (Figure 2). The gameplay starts with the user selecting from the dropdown list box (Figure 3) a mission. The user then creates a flowchart for the mission, and finally tests the algorithm by executing the flowchart which is presented by the robot's movement.

At the initialisation of every flowchart, the robot has a set amount of energy to run the course. If the energy runs out before the algorithm can complete, then the robot stops and the mission is a fail. The robot can be recharged during gameplay by a socket, if available on the ground.

Figure-2: New file created in RobotProg

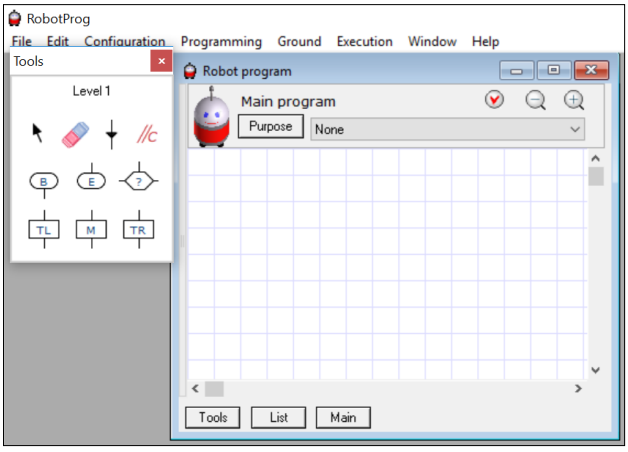
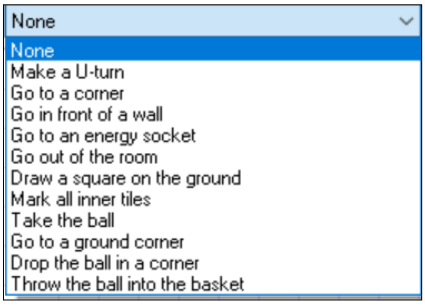


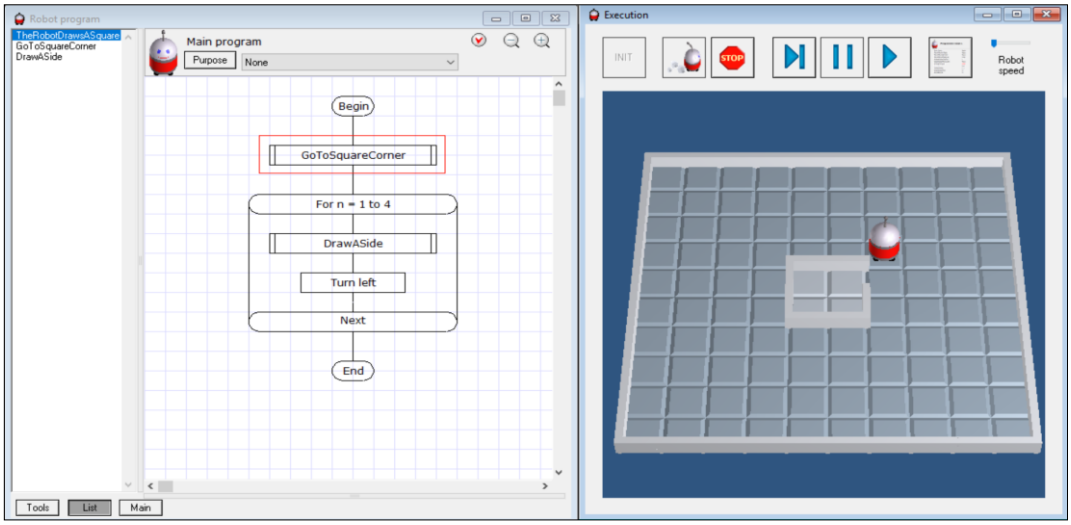
Figure-3: List of missions



Source: RobotProg: 2005: Screen Capture (SC). *Source*: RobotProg: 2005: SC.

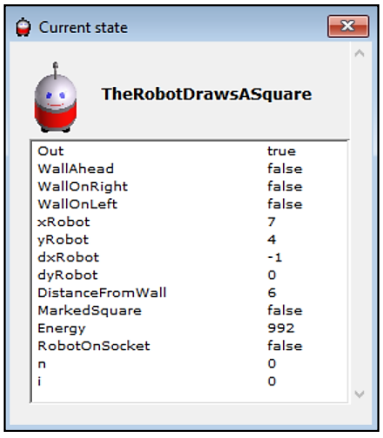
During the execution of the flowchart, the ground window is shown alongside and the symbol that is currently being executed is outlined in red (Figure 4, next page). Also, the current state (Figure 5, next page) shows a live update of the robot's actions and variables. If the mission is successful, the user is applauded with a sound effect and proceeds to a new mission until the next level is reached.

Figure-4: Red outline during the execution of the flowchart



Source: RobotProg: 2005: SC.

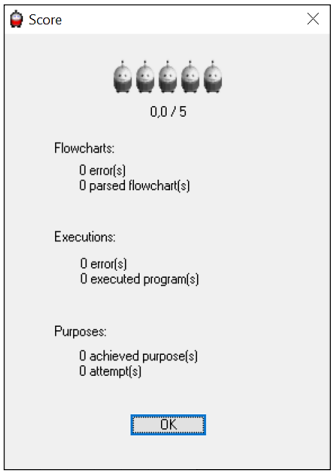
Figure-5: Current state



Source: RobotProg: 2005: SC.

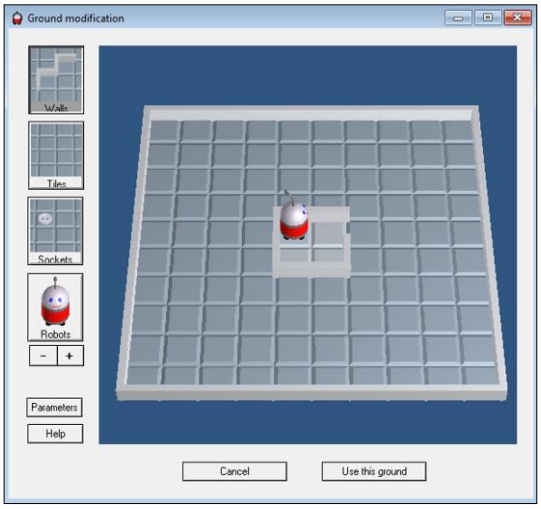
The built-in scoreboard keeps a record of mission status, number of flowchart errors, attempts and successful executions (Figure 6). Different missions of the robot are based on different grounds, the user has the liberty of creating a custom ground (Figure 7) on which the robot navigates to complete the mission. Each gameplay allows up to ten robots each having a flowchart that is added to the ground and energy to perform the required task. In addition to the standard flowchart symbols, there are special additional symbols added to control the robot such as TL-turn left, TR-turn right, M-move ahead, etc. and specific keywords that can be understood with minimal effort since they have a literal meaning.

Figure-6: Scoreboard



Source: RobotProg: 2005: SC.

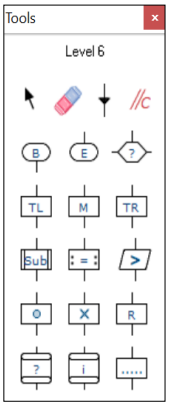
Figure-7: Ground design



Source: RobotProg: 2005: SC.

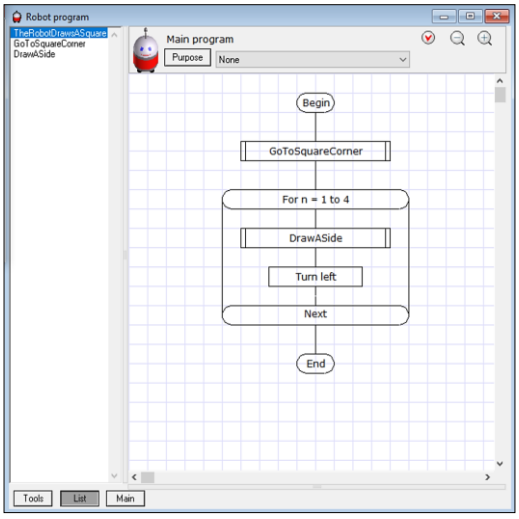
Each level unlocks different flowchart symbols which are useful to complete the level mission at hand (Figure 8, next page). RobotProg offers six levels with each level unlocking the discovery of new programming concepts. Progressive learning development is demonstrated, requiring the individual to draw on past knowledge and the newly available symbols to achieve the mission at hand. This draws on the Zone of Proximal Development in the learning process. At level six the user has liberal access to all available flowchart symbols and has acquired the skill set to utilise the symbols that would develop an efficient algorithm. In the flowchart titled *The Robot Draw A Square* (Figure 9, next page), we see the use of subroutines: Go To Square Corner (Figure 10, next page) and Draw A Side.

Figure-8: Flowchart symbols at level six



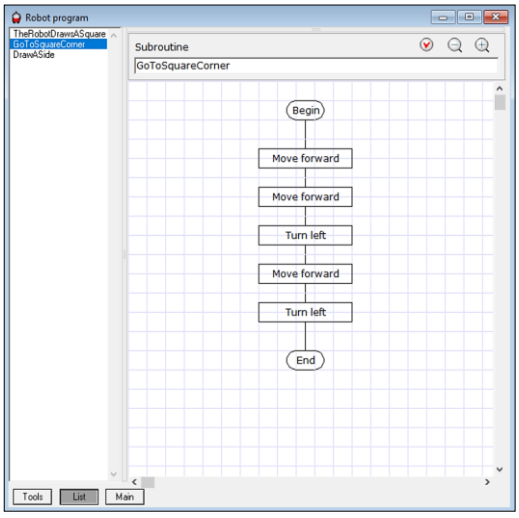
Source: RobotProg: 2005: SC.

Figure-9: The robot draw a square



Source: RobotProg: 2005: SC..

Figure 10: Go To Square Corner



Source: RobotProg: 2005: SC.

The use of subroutines in RobotProg serves the purpose as in a text-based environment, that is having an algorithm to perform a specific task and called up when needed. This introduces the ESP to modular design and re-use of code leading to advanced concepts later explored in a text-based environment.

4. RESULTS

RobotProg effectively addresses the factors of Fogg's Behaviour Model as it triggers motivation and ability which results in an efficient problem solver. The desired behaviour is reached during each mission. Learning takes place while the individual is motivated to accomplish the mission, gains the ability through the use of available symbols and past knowledge; and is triggered to move onto the next level. Successful completion of missions results in level progression and ground modification. The latter allows for obstacles to be added to the robot's course, thus requiring careful and strategic planning in the design of the flowchart, thereby creating a challenge for the ESP.

Table 1 shows the evaluation of RobotProg, using the checklist of *Tutorial-based evaluation on user perspective* adapted from Jackson, Crouch & Baxter (2011).

Table 1: Checklist for user perspective versus RobotProg.

Criteria for user perspective	Criteria met (✓) OR not met (✗)
Visibility of system status. Does it give users appropriate feedback within reasonable time?	✓ Simple fast installation with confirmation.
Does it speak the user's language and make information appear in a natural and logical order?	✓ Easy to understand instructions in English with clear logical order of panels and windows.
User control and freedom. Does it provide clearly marked exits, undo and redo?	✓ Icons appear on menu bar.
Consistency and standards. Is it consistent within the software?	✓ Common theme used.
Does it prevent errors in the first place or help users avoid making them?	✓ Prevents incorrect linking of flowchart symbols.
Does it make objects, actions and options visible and reduce the amount of information a user has to remember?	✓ Use of icons appear together with Labels.
Flexibility and efficiency of use. Does it offer short cuts?	✓ Shortcuts are available instead of accessing via the menu bar.
Aesthetic and minimalist design. Does it avoid showing irrelevant information?	✓ Windows/panels only appear when needed.
Help users recognize, diagnose, and recover from errors. Does it make errors clear, comprehensible, precise and suggest solutions if possible?	✓ If there is an error the robot does not complete the mission and this is accompanied with a sound effect.
Help and documentation. Does it provide concise information?	✓ Help tab available on menu bar with detailed explanation.

RobotProg meets all the criteria by the *Tutorial-based evaluation on user perspective* (Table 1). Meeting this criteria ensures a pleasing user experience while being motivated to learn. The design of the RobotProg promotes programming concepts like the set energy of the robot at initialisation demonstrates programming efficiency. As one transits to a specific programing

language, one would be able to design algorithms that do not become monotonous and heavy laden on processing time. RobotProg provides information about variables and actions of the robot during the execution of code which mimics the steps carried in a trace table. Therefore the ESP, not only learns to debug flowcharts by spotting errors but eliminates redundant steps. This process nurtures and develops the mental state of the new programmer by invoking metacognition while developing a programmer's mindset. As a result there is a level of convincing that is attained before the transition to a programming language.

5. DISCUSSION

Flowcharts are considered a visual aid to programming with the use of symbols. RobotProg allows for the flowchart to come alive, thus enhancing the students' understanding visually. The instructional tool RobotProg allows the design of the flowchart, execution of the flowchart and monitors the movements of the robot through gameplay. The findings of RobotProg, a flowchart-based programming environment, and the potential impact in aiding ESPs are listed below:

Whenever a flowchart is executed, the student is exposed to the actions of the flowchart through the movement of the robot, contributing to an increased focus on problem-solving skills, and the vital concepts of programming with no burden on syntax errors. Constructs such as loops, conditions and modular programming, through the use of subroutines, allow the programmer to acquire advanced conceptual knowledge in a visual environment, before moving to a text-based environment.

The automated feedback through sound and robot animation in a gamified environment have the potential to improve the learning experience and problem-solving skills of ESPs as it is engaging and fun. The tool offers a visual simulation in learning flowcharts that can significantly help improve problem-solving skills and program composition modeling.

At an early programming stage, the software offers simultaneous execution of the algorithm allowing for step by step execution of flowcharts with simultaneous inspection of variables. This leads to debugging concepts like trace tables. The welcoming of multiple solutions can be easily determined by completed missions.

The tool is understood without any special requirements or training. The focus is directed towards the foundations of programming: iteration, sequence and selection structures. Emphasis is placed on program composition and execution

flow. The ESP directs their focus towards problem-solving, develops their knowledge base of programming concepts and techniques while minimising the influence of complicated programming languages.

Students will have greater confidence in using programming concepts via RobotProg and later in a text-based environment, rather than flowcharts in a static pedagogic environment. The graphical user interface of RobotProg provides features like *initialisation* and *run* so the ESP can experience the layout of an Integrated Development Environment (IDE).

The introduction of flowcharts in this manner results in the skill of debugging and the eradication of misconceptions since the output of flowcharts is rendered through the robot's actions. The motivation to learn flowcharts and the trigger to persevere into text-based environments are embraced by RobotProg.

6. CONCLUSION

Through the reflective overview, it is suggested that there are benefits for ESPs to use RobotProg flowcharts moving away from the dull superannuated learning style of pen and paper. This is because it has the potential to emphasise logic and code design in the first programming course inducing motivation and excitement through gameplay. The completed example projects, make us believe that the environment created by the software offers self-facilitated learning of flowcharts.

ACKNOWLEDGEMENT

This work is based on the research supported by the National Research Foundation of South Africa (Grant Number: 122017).

REFERENCES

Buckley, P. & Doyle, E. (2016). Gamification and student motivation. *Interactive learning environments*, 24(6), 1162-1175.

Burton, B. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4(1), 3-14.

Cabo, C. (2018). Effectiveness of Flowcharting as a Scaffolding Tool to Learn Python. In Proceedings of the 49th IEEE Frontiers in Education Conference (1-7). Cincinnati Marriott at RiverCenter, Ohio.

Chapin, N. (1970). Flowcharting with the ANSI standard: A tutorial. *ACM Computing Surveys (CSUR)*, 2(2), 119-146.

Chen, S. & Morris, S. (2005). Iconic programming for flowcharts, java, turing, etc. *ACM SIGCSE Bulletin*, 37(3), 104-107.

Collins, F. (2017). *SA gaming grows from R29.7-million to R100-million industry in two years.* https://www.timeslive.co.za/news/sci-tech/2017-06-08-sa-gaming-grows-from-r297-million-to-r100-million-industry-in-two-years/

Futschek, G. & Moschitz, J. (2010). Developing algorithmic thinking by inventing and playing algorithms. In Proceedings of the 2010 Constructionist Approaches to Creative Learning, Thinking and Education Lessons for the 21st Century (1-10). Paris, France.

Fogg, B. J. (2009). A behavior model for persuasive design. In Proceedings of the 4th international Conference on Persuasive Technology (1-7). California, United Sates of America.

Gul, S., Asif, M., Ahmad, W. & Ahmad, U. (2017). Teaching programming: A mind map based methodology to improve learning outcomes. In 2017 International Conference on Information and Communication Technologies (209-213). Karachi, Pakistan.

Hsu, C. C. & Wang, T. I. (2018). Applying game mechanics and student-generated questions to an online puzzle-based game learning system to promote algorithmic thinking skills. *Computers & Education*, 121(1), 73-88.

Huotari, K. & Hamari, J. (2017). A definition for gamification: Anchoring gamification in the service marketing literature. *Electronic Markets*, 27(1), 21-31.

Knuth, D. E. (1963). Computer-drawn flowcharts. *Communications of the ACM*, 6(9), 555-563.

Lehman, M. W. (2000). Flowcharting made simple. *Journal of Accountancy*, 190(4), 77.

Lo, C. K., & Hew, K. F. (2018). A comparison of flipped learning with gamification, traditional learning, and online independent study: the effects on students' mathematics achievement and cognitive engagement. *Interactive Learning Environments*, 2018(1), 1-18.

Morris, C. (2018). *Spending on Video Games Increases 40% in the First Half of 2018*. http://fortune.com/2018/08/29/video-games-spending-increases-2018/

Myler, H. R. (1998). Fundamentals of engineering programming with C and Fortran. New York: Cambridge University Press.

Ray, S., Herrera-Cámara, J. I., Runyon, M. & Hammond, T. (2019). Flow2Code: Transforming Hand-Drawn Flowcharts into Executable Code to Enhance Learning. In T. Hammond, M. Prasad & A. Stepanova (Eds.) *Inspiring Students with Digital Ink* (pp.79-103). Switzerland: Springer.

Scanlan, D. A. (1989). Structured flowcharts outperform pseudocode: An experimental comparison. *IEEE software*, 6(5), 28-36.

Shneiderman, B., Mayer, R., McKay, D. & Heller, P. (1977). Experimental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM,* 20(6), 373-381.

Singh, A. K. (2020). Computational Science. India: BlueRose Publishers.

Tarentaal, D. (2019). *How SA can grow its gaming industry: Price Water House Coppers report identifies the digital video games sector as one of SA's 'biggest success stories' in the entertainment and media industries.* https://www.moneyweb.co.za/news/tech/how-sa-can-grow-its-gaming-industry/

Turner, A. (2006). Flowcharts for a Smooth Ride. *Quality progress*, 39(7), 96.

Wijman, T. (2018). *Mobile Revenues Account for More Than 50% of the Global Games Market as It Reaches $137.9 Billion in 2018*. https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/